Pattern Recognition 48 (2015) 3993–4011

Contents lists available at ScienceDirect

Pattern Recognition

journal homepage: www.elsevier.com/locate/pr

A generalisable framework for saliency-based line segment detection

Mark Brown *, David Windridge, Jean-Yves Guillemaut

Centre for Vision, Speech and Signal Processing, University of Surrey, Guildford, Surrey GU2 7XH, UK

ARTICLE INFO

Article history:
Received 26 January 2015
Received in revised form
14 May 2015
Accepted 25 June 2015
Available online 6 July 2015

Keywords:
Line detection
Feature detection
Feature matching
Registration
Saliency

ABSTRACT

Here we present a novel, information-theoretic salient line segment detector. Existing line detectors typically only use the image gradient to search for potential lines. Consequently, many lines are found, particularly in repetitive scenes. In contrast, our approach detects lines that define regions of significant divergence between pixel intensity or colour statistics. This results in a novel detector that naturally avoids the repetitive parts of a scene while detecting the strong, discriminative lines present. We furthermore use our approach as a *saliency filter* on existing line detectors to more efficiently detect salient line segments. The approach is highly generalisable, depending only on image statistics rather than image gradient; and this is demonstrated by an extension to depth imagery. Our work is evaluated against a number of other line detectors and a quantitative evaluation demonstrates a significant improvement over existing line detectors for a range of image transformations.

1. Introduction

Line segments are an important low-level feature, particularly where man-made structures are present. In many situations they may be used in a similar manner to points, e.g. pose estimation [5], stereo matching [9], or structure from motion [8]. This may often be helped by using the *duality* between lines and points, resulting in similar registration approaches for the two types of feature [26]. Further, there are tasks especially suited to lines, e.g. vanishing point estimation for camera calibration [10], image resizing [17], or structural graph matching [19].

Existing line detection methods either first use a derivative-based edge detector and detect lines from the edges (e.g. [4] or via the Hough Transform [6]), or they directly group pixels in the image into line segments based on the magnitude and direction of their derivative [49,14]. However, these all act locally on the image, detecting a large number of lines, particularly in repetitive scenes. This limitation is illustrated[1] in Fig. 1: state of the art line detection detects all lines regardless of their significance, whereas, ideally, the non-repetitive lines denoting the geometrically significant edges would be preferentially detected.

To address this, we propose to detect only the *salient* line segments, an area that, to the best of the authors' knowledge, has not been addressed in the literature. Instead, saliency detection commonly refers to the computation of a *saliency map* (e.g. [31]), with some work addressing salient edge detection [28] and salient point detection [32]. In detecting only the salient line segments, we propose an approach that is fundamentally different from existing methods for line segment detection in that it is not derivative-based: instead, it seeks informational contrast between regions and thereby favours non-repetitive edges. The information is expressed in terms of distributions of pixel intensities taken from rectangles of a variable width, meaning our approach operates over a larger scale than other detectors and so naturally avoids repetitive parts of a scene.

We measure the contrast between the two distributions on either side of the line using the information-theoretic Jensen–Shannon Divergence (JSD). This measure has been used elsewhere for edge detection [39], unlabeled point-set registration [50], and DNA segmentation [25]. It has many interpretations, e.g. it may be expressed in terms of other information-theoretic quantities such as the Kullback–Liebler Divergence and Mutual Information, having further interpretations in both statistical physics and mathematical statistics [25], and is the square of a metric.

Our measure of line saliency may further be used as a *saliency filter* on existing line detectors. This allows it to cull the non-salient line segments computed by other detectors and localise the position of salient lines under our saliency measure. It furthermore increases the speed of salient line detection by orders of magnitude over the naive approach of determining the saliency measure of every possible line segment on the image.

This distribution-based approach to line detection we propose is highly generalisable, being applicable to any situation where informational contrast can be found. As such, we implement an extension for line detection in depth images, whereby lines that

* Corresponding author. Tel.: +44 1483 686046.
E-mail addresses: m.r.brown@surrey.ac.uk (M. Brown), d.windridge@mdx.ac.uk (D. Windridge), j.guillemaut@surrey.ac.uk (J.-Y. Guillemaut).

[1] Image by Gila Brand, http://commons.wikimedia.org/wiki/File:Washington_DC_windows.jpg. Licensed under CC BY 2.5. Grayscale of original.

http://dx.doi.org/10.1016/j.patcog.2015.06.015

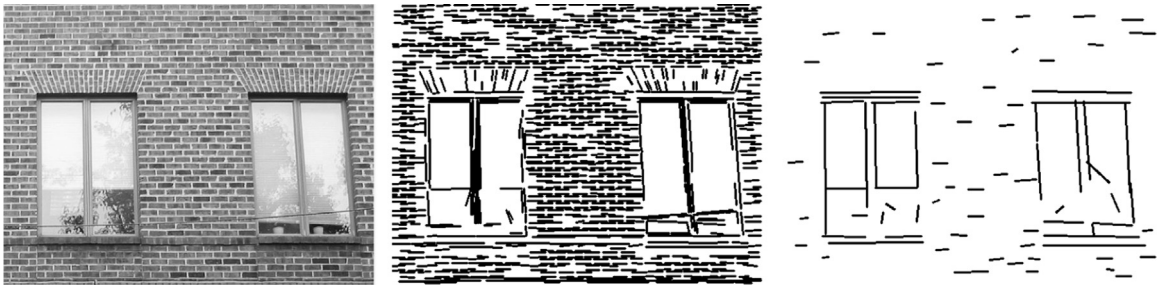

Fig. 1. Left: Input image. Centre: LSD algorithm, [49], returning 1026 lines. Right: Our proposed approach, returning 75 lines, indicative of the broad structure of the scene.

jointly delineate changes in surface orientation or texture are detected. These are reprojected, allowing for 3D salient line detection and hence potential multi-modal applications.

The contributions of this paper are as follows: firstly, a distribution-based salient line segment detector is formulated and implemented: the first known method for salient line segment detection. Secondly, the notion of *saliency-based filtering* is applied to existing line detectors for efficient salient line detection. Thirdly, an extension to depth imagery is implemented, allowing for the detection of salient lines in 3D structures. An evaluation shows that, when considering that we detect only a small number of lines, our approaches significantly outperform the others in terms of repeatability and homography estimation. It demonstrates that they are representative of the underlying aspects of the scene, with potential use for problems that benefit from fewer, but more reliable, features e.g. [20].

The structure of the paper is as follows: in Section 2, we review related work in line detection, edge detection, and line detection in depth imagery. In Section 3 the methodology is described for both salient line detection and saliency filtering, with the extension to depth imagery (and subsequently 3D by reprojection) described in Section 4. In Section 5 a range of qualitative and quantitative results are given, and in Section 6 our conclusions and ideas for future work are presented.

2. Related work

Since we are unaware of any research into salient line detection (or any line detection method that does not act locally on the derivative of the image) we firstly review line segment detection, before reviewing relevant edge detection methods. Finally, line detection in other modalities (depth images, 3D data) is reviewed.

2.1. Line detection

Most early methods of line detection relied upon the Hough Transform (HT) [6] to determine a set of lines from a set of edges (typically extracted from the image by the Canny edge detector [15]). The HT exhaustively searches the space of all possible infinitely long lines, determining how many edge pixels are consistent with each line; lines with a suitably large number of edge pixels lying on them are returned as the output of the algorithm. In its naive form there are many drawbacks, for example it only depends on the magnitude of the gradient and not the orientation, and leaves a problem of how to accurately determine the endpoints of the lines. However, there are many variants of the Hough Transform [30] that seek to solve some of these problems.

Regardless of the approach to line detection, early methods particularly suffered from the problem of setting meaningful thresholds. This was addressed by the Progressive Probabilistic Hough Transform (PPHT) [41] by Matas et al. where it is achieved in a probabilistic manner: the threshold is expressed in terms of the probability of the line occurring by chance. The idea was extended by Desolneux et al. [21] who exhaustively search every line segment on the image and define an *a contrario* model to control the number of false detections. The latter part is a straightforward extension: if there are N possible line segments on an image and p is the probability of that line segment occurring by chance, then accepting the line if $p < \epsilon/N$ guarantees, on average, ϵ false detections per image.

However, Grompone von Gioi et al. [48] note that this model, in its current form, is too simple. Given an array of line segments, the model tends to interpret it as one long line, leading to unsatisfactory results. This is not a fault of the *a contrario* model, but rather that it is applied to each line individually. If instead it is applied to groups of lines at a time it will segment a line into its components in the correct manner, known as 'multi-segment analysis'. However, this adds another layer of complexity, becoming $O(N^5)$ for an $N \times N$ image.

Grompone von Gioi et al. subsequently implemented a linear-time Line Segment Detector (LSD) [49]. It is based on both the *a contrario* model and an earlier line detection algorithm by Burns et al. [14]. It is a spatially based approach, starting from small line segments and growing them. Furthermore, each segment has its own *line support region*, constructed by grouping nearby pixels that have a similar gradient, thus detecting lines of variable width.

The *a contrario* model has also been implemented in the EDLines detector by Akinlar and Topal [4]. The approach performs similarly to LSD but up to ten times faster due to its very fast edge detection algorithm that simultaneously detects edges and groups them into connected chains of pixels. Less processing time is required for subsequent line detection, resulting in a real-time line segment detector.

All line detection methods reviewed above are unable to detect lines based on their significance or surroundings. Consequently, they tend to return a large number of lines which does not capture the general structure of the scene.

2.2. Edge detection in images

Similarly to approaches to line detection, many approaches to edge detection act locally on the image. One of the earliest algorithms, the Canny edge detector [15], convolves the image with a Gaussian filter before computing the magnitude of the gradient at each pixel. Variants have been proposed in particular for the convolution stage; notably Liu and Feng [38] use an anisotropic Gaussian filter that only operates perpendicularly to an edge. It is combined with a multi-pixel search to detect longer edges than other approaches, culminating in the detection of short edge-line segments. Their results indicate superior performance compared to existing edge detectors in the presence of different levels of Gaussian noise. However, both approaches are fundamentally derivative based, acting locally on the image regardless of the structure of the scene.

In contrast there exist some non-local edge detection methods. For example, Holtzman-Gazit et al. [28] determine salient edges by combining an edge preserving filter with a regional saliency measure in a multi-scale manner. It complements existing approaches to salient feature detection, e.g. [32,46] where salient point detection is formulated in an information theoretic manner. Other approaches to edge detection are more similar to ours in that they are distribution-based, in fact, the JSD has already been used for edge detection [39] via a sliding window approach. For each pixel and each orientation, the JSD is evaluated between a distribution from one side of the pixel and the other. However, the method is not scale-invariant, with the sliding window being a fixed size throughout the algorithm. Further parameters have not been discussed e.g. how to determine a probability distribution from a sliding window, and the algorithm is only tested on one image.

An advantage of distribution-based edge detection is its natural extension to colour images, as Ruzon and Tomasi [45] do. They use the Earth Mover's Distance (EMD) between distributions which formulates distance as a transportation problem; it represents the minimum amount of work to 'transport' one distribution into the other. Their method obtains good qualitative results but is very time consuming (about 10 min per image) and furthermore is not scale-invariant. The distribution-based approaches to edge detection reviewed here have proved promising, however they have never been used for the detection of salient straight lines.

Edge detection is a very important low-level operation with numerous applications. For example, the HT [6] may be used on an edge map to detect line segments, however this has its own limitations e.g. it does not take into account the orientation of the edges detected. Edges may be locally chained into line segments directly [23] however the approach detects many false positives since a meaningful threshold has not been set. Computing a polygonal approximation to a contour (i.e. a connected set of edge pixels) will determine a representative set of lines for the curve. A range of algorithms have been proposed [3,11,43], for example Parvez [43] relaxes a condition that vertices of the approximating polygon need to lie on the contour, and Bhowmick and Bhattacharya [11] relax the definition of a digitally straight line: both of these allow for a polygon approximation formed by fewer, more meaningful line segments. Bhowmick and Bhattacharya [11] furthermore propose a very quick algorithm relying only on primitive integer computations; efficient algorithms have also been proposed by Aguilera-Aguilera et al. [3] who use a concavity tree to more quickly determine the vertices of the polygon approximation. However, polygon approximation algorithms inherently approximate curves by a set of lines, in contrast to our approach to detecting straight lines that avoids curved segments completely.

2.3. Line detection in other modalities

In the 3D or depth imagery domain, there has been much research on edge detection (e.g. [44]) with relatively little on straight line detection.

Some research focuses on detecting lines in textureless 3D data. For example, Stamos and Allen [47] detect planes in 3D and determine lines as the intersection of these planes. Lin et al. [37] convert a 3D model into a set of shaded images using a non-photorealistic rendering technique that provides a strong perception of edges. 2D lines are detected on the shaded images using the LSD algorithm [49] and reprojected to 3D where a line support region is constructed. However, both approaches detect lines based purely upon the geometry of the 3D scene. With respect to textured 3D data, Chen and Wang [18] detect lines in 3D point clouds that have been reconstructed by Structure-from-Motion (SfM) by detecting and reprojecting lines in the images used to generate the point cloud. It is difficult to apply this approach to general 3D point clouds without manually creating a set of camera locations with which to detect 2D lines from. Buch et al. [13] extend the RGB-D edge detection method presented in [44] by approximating each edge by a line segment. However, the length of the line segment is a parameter of the algorithm, meaning all detected line segments have the same length (in the image space).

The key novelty of our contribution lies in the distribution-based approach to line detection that is proposed, resulting in a method that naturally avoids the repetitive parts of a scene and returns only the salient line segments present. This generalisable approach allows for natural extensions into other modalities, which is demonstrated via a depth extension. We are not aware of any other methods that explicitly detect straight lines in depth images in such a way that jointly delineates changes in surface orientation or texture. Additionally, saliency *filtering* is proposed to cull non-salient line segments obtained from other approaches, thereby achieving significantly faster processing times.

3. Methodology

The methodology can be broadly split into two stages. The *first stage* searches all possible lines on the image, calculating the saliency value (S_{val}) of each line, and accepting it according to a certain set of conditions. To do so requires the estimation of the Jensen–Shannon Divergence (*JSD*) between two sets of data, as outlined in Sections 3.1 and 3.2 details the first stage: how the *JSD* relates to the saliency value of a line and how to compute a putative set of lines from this. In the *second stage* the most *representative* set of lines is determined from this putative set, as outlined in Section 3.3. This is achieved using affinity propogation [24], a fast clustering algorithm that works particularly well for larger numbers of clusters. Hence the resulting algorithm returns a representative set of lines for the scene. A flowchart of the algorithm, along with intermediate results from each section, are shown[2] in Fig. 2.

The above algorithm has a high complexity (for an $N \times N$ image it has $O(N^5)$ complexity) because it performs an exhaustive search over all lines at all scales in an image. Therefore in Section 3.4, we propose an alternative approach: a *saliency filter* on top of existing line detectors that filters out non-salient lines. This allows it to detect only the salient lines within seconds: orders of magnitude faster than the above approach.

3.1. Estimating the Jensen–Shannon divergence

Let P and Q be discrete probability distributions, taking one of K values (i.e. $P = \{p_1, \ldots, p_K\}, p_i \geq 0 \ \forall i, \sum_{i=1}^{K} p_i = 1$). The *entropy* of a probability distribution is defined as

$$H(P) = -\sum_{i=1}^{K} p_i \ln p_i \tag{1}$$

The *JSD* between two probability distributions P and Q is subsequently defined as

$$JSD(P,Q) = H\left(\frac{P+Q}{2}\right) - \frac{H(P)+H(Q)}{2} \tag{2}$$

It is closely related to other information-theoretic quantities such as the Mutual Information (*MI*) or the Kullback–Liebler Divergence (*KLD*) [25] and shares similar properties and interpretations.

[2] Image by Russ Hamer, http://commons.wikimedia.org/wiki/File:Melton_-Mowbray_St_Marys_SE_aspect.JPG. Licensed under CC BY-SA 3.0. Grayscale of original.

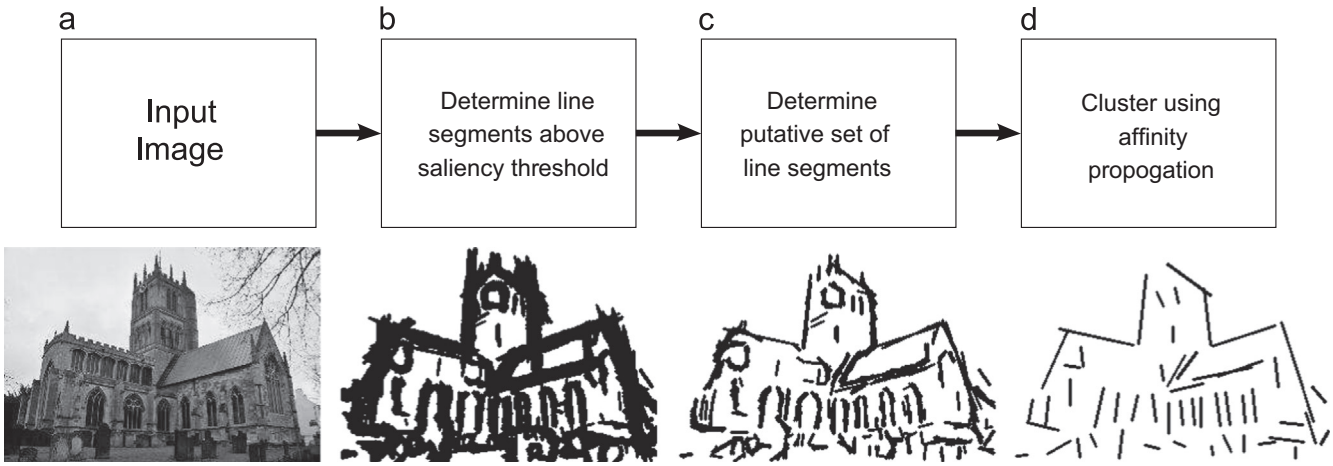

Fig. 2. Top: Pipeline of the algorithm. Bottom: an illustration of the effects of the different steps of the algorithm. (a) Input image. (b) All line segments above S_{thresh} (6,480,144 segments returned). (c) Putative set of line segments (above S_{thresh} subject to the three further principles) (603 segments returned). (d) All line segments after affinity propogation (52 segments returned).

Indeed, for discrete random variables M and Z the MI is defined as

$$I(M,Z)=\sum_{m\in Mz\in Z}\sum Pr(m,z)\ln\left(\frac{Pr(m,z)}{Pr(m)Pr(z)}\right) \tag{3}$$

where $Pr(m,z)$ denotes the joint probability of the event $(M=m,Z=z)$ and $Pr(m)$, $Pr(z)$ are the *marginal probabilities* of the events $(M=m)$ and $(Z=z)$ respectively. Then, if M is the random variable associated with the distribution $P+Q/2$ and Z is a binary indicator variable denoting which of P or Q a sample of M was generated from, one sees that, by a little algebraic manipulation [25], $JSD(P,Q)=MI(M,Z)$.

The MI has an information theoretic interpretation: it represents the average number of extra *nats* (bits taken to base e) that need to be transmitted to encode the product distribution $Pr(M,Z)$ from a code using only the marginal distributions. Subsequently, the relationship between the JSD and MI shows that $JSD(P,Q)$ is bounded between 0 (when P and Q are the same) and ln 2 (when P and Q are completely different). Grosse et al. [25] give further statistical interpretations of $JSD(P,Q)$ and Endres and Schindelin [22] show it is the square of the metric.

In reality, one is never able to exactly know the distributions P and Q, and instead it must be estimated from samples of data. Assume there are N samples of data from both P and Q with counts represented by $\mathbf{n}=\{n_1,\ldots,n_K\}$ and $\mathbf{m}=\{m_1,\ldots,m_K\}$, respectively, hence $\sum_{i=1}^{K}n_i=\sum_{i=1}^{K}m_i=N$. Then $JSD(P,Q)$ may be estimated by calculating the observed JSD: $JSD_{obs}(\mathbf{n},\mathbf{m}):=JSD(\mathbf{n}/N,\mathbf{m}/N)$. However, this is only an estimate, and it suffers from two important limitations.

Firstly, there is a *systematic bias* in this naive approach (see [25]), with JSD_{obs} expected to be higher than the JSD of the true, underlying distribution of pixel intensities (JSD_{true}). This is particularly evident when P and Q are uniform distributions: JSD_{true} is zero but measurements $\mathbf{n}$ and $\mathbf{m}$ will most likely cause JSD_{obs} to be non-zero. The bias is particularly large when N is small, and tends to zero as the sample size becomes arbitrarily large. Furthermore, when N is small, there is a high probability that a given value of JSD_{obs} could have occurred by chance, and this needs to be reflected in any estimate of JSD_{true}.

Both of these problems may be solved by computing a Bayesian estimate of JSD_{true}. This directly avoids the problem of systematic bias, and naturally accounts for smaller sample sizes N by assuming and integrating over a symmetric *Dirichlet prior*. The Dirichlet prior defines a *prior probability* for distributions P (respectively Q) by a parameter α as $\mathbb{P}(p_1,\ldots,p_K;\alpha)\propto\prod_{i=1}^{K}p_i^{\alpha-1}$. It is a general prior since it is parameterised by α: $\alpha=1$ corresponds to a uniform prior, $\alpha=0.5$ to Jeffrey's prior, etc. Informally, the magnitude of α corresponds to the size of the prior, with larger values of α representing a large prior belief that P (resp. Q) is evenly distributed. With this prior, the Bayesian estimate for JSD_{true} is defined as follows:

$$JSD_{est}(\mathbf{n},\mathbf{m}):=\mathbb{E}(JSD(P,Q)\,|\,\mathbf{n},\mathbf{m},\alpha) \tag{4}$$

$$JSD_{est}(\mathbf{n},\mathbf{m})=\int_{X\in\Omega}\int_{Y\in\Omega}JSD(X,Y)\mathbb{P}(X\,|\,\mathbf{n},\alpha)\mathbb{P}(Y\,|\,\mathbf{m},\alpha)\,dX\,dY \tag{5}$$

where the integral is taken over the space of all probability distributions Ω ($\Omega=\{\omega_1,\ldots,\omega_K\}$, $\omega_i\geq 0\forall i$, $\sum_{i=1}^{K}\omega_i=1$).

We employ the results of Hutter [29] who calculates a Bayesian estimate of the MI between two random variables M and Z given a finite set of samples, and then modify his solution for the JSD. The result for the MI is as follows: firstly, denote $s'_{m,z}$ as the number of samples taking the joint value (m,z) and let $s_{m,z}=s'_{m,z}+\alpha$. Denote marginal sums $s_{m,+}=\sum_{z\in Z}s_{m,z}$ and $s_{+,z}=\sum_{m\in M}s_{m,z}$ and let $s_{++}=\sum_{z\in Z}\sum_{m\in M}s_{m,z}$. Then Hutter computes the Bayesian estimate as

$$\mathbb{E}(I(M,Z)\,|\,s)=\frac{1}{s_{++}}\sum_{m\in Mz\in Z}\sum s_{m,z}[\psi(s_{m,z}+1)-\psi(s_{m,+}+1)-\psi(s_{+,z}+1)+\psi(s_{++}+1)] \tag{6}$$

where ψ is the digamma function defined as $\psi(x)=\Gamma'(x)/\Gamma(x)$. As previously stated, $JSD(P,Q)$ may be reformulated in terms of $I(M,Z)$, where M represents the mixture distribution $P+Q/2$ and Z is a binary indicator variable denoting which of P or Q a sample of M was generated from. Using these substitutions, Hutter's result may be re-written to compute a Bayesian estimate of $JSD(P,Q)$ given a finite set of samples $(\mathbf{n},\mathbf{m})$ as follows:

$$JSD_{est}(\mathbf{n},\mathbf{m})=\frac{z(\mathbf{n})+z(\mathbf{m})-z(\mathbf{n}+\mathbf{m}+\boldsymbol{\alpha})}{2(N+\alpha K)}+\psi(2(N+\alpha K)+1)-\psi(N+\alpha K+1) \tag{7}$$

where we define $z(\mathbf{x}):=\sum_{i=1}^{K}(x_i+\alpha)\psi(x_i+\alpha+1)$ and $\boldsymbol{\alpha}$ is a K-vector of all α's. Note that Eq. (7) applies directly to the data $\mathbf{n}$ and $\mathbf{m}$ and hence may be computed with similar efficiency to the naive JSD_{obs} computation. Hence, small sample problems and systematic bias may be naturally avoided as easily as directly computing the observed JSD.

3.2. Computing a putative set of lines

The saliency of a line segment can be related to the previous section in the following way. Suppose a line segment L has start and end points at (x_1, y_1) and (x_2, y_2) with length $\|L\|$. Let T be a rectangle adjoining L lying on one side of L and B a rectangle on the other side. We define the scale, s, of the line to be the width of each rectangle, where s is allowed to vary, taking any value up to $\|L\|$ (See the left of Fig. 3 for an illustration). Subsequently, $\mathbf{n}$ represents a histogram of pixel intensities from T and respectively $\mathbf{m}$ from B. Denote the estimated JSD between the two regions as $J(L,s) := JSD_{est}(\mathbf{n}, \mathbf{m})$, calculated according to the previous section. The left of Fig. 4 shows two typical lines, their rectangles and their histograms. Particularly evident in this image is how similar distributions either side of a line in a repetitive structure (e.g. brickwork) are, and hence how these are naturally avoided by our approach.

A significant problem with simply using the estimated *JSD* of regions taken from either side of the line as a saliency measure is its poor localisation. For example, let L be a line whose $JSD(L,s)$ value is particularly high. Then any line $L' \subset L$ also has a high estimated *JSD* value, making it very difficult to determine the endpoints of L.

We observe however that *beyond* the endpoints of a line JSD_{est} should be very low (since, by definition, there is no line there). Thus, let L_L and L_R denote line segments taken from beyond left and right endpoints of L, with adjoining rectangles T_L, B_L, T_R and B_R, respectively (see the right of Fig. 3). Motivated by the desire to keep $J(L,s)$ high and $J(L_L,s)$ and $J(L_R,s)$ low, we use the following measure of line saliency:

$$Sal(l,s) = J(L,s) - \beta(J(L_L,s) + J(L_R,s)) \tag{8}$$

where β is empirically determined to be 0.25. The right of Fig. 4 shows an example of a localised line segment, where the distributions on either side of the line *beyond* its endpoints are similar, but are very dissimilar on either side of the segment itself.

A formulation of line saliency has now been defined, allowing for the detection of salient lines under this saliency measure. In a similar manner to [21], our measure of line saliency is evaluated on all lines of the image by first evaluating it on all horizontal lines of the image with pixel level precision (i.e. the start and end points of the line are integer valued with the same y-coordinates). The process is repeated on r evenly spaced rotations of the image. However, computing all lines on an image whose saliency value is higher than a given threshold does not, on its own, give meaningful results. Fig. 2(b) demonstrates this: over 6 million segments are returned across multiple scales—far too many to be of practical value. Three further principles are employed in order to decide whether to accept a given line segment:

- *Local Maxima*: The line has to be more salient than its immediate neighbours. The neighbours are in five dimensions corresponding to the scale s and the coordinates of the two endpoints of line L.
- *Maximally Salient*: Maximal saliency is defined as in [21]: a line segment is *maximally salient* if it does not contain a strictly more salient segment *and* it is not contained in a more salient segment.
- *Scale Selection*: Many feature detectors (e.g. [40,32]) search for features that have a high response value at a particular scale. However, we observe for our case that lines that are salient across a range of scales are more desirable. Consider a line segment along a jagged edge (e.g. Fig. 3). It may have a high saliency value $Sal(l,s)$, particularly so if s is large, however for small s the line is not salient due to the jagged nature of the line. Kadir and Brady [32] note this is an appropriate approach to scale selection for edges since there is no associated scale in their tangential direction. Hence, we introduce a lower threshold $J_{\min}$ and only accept lines of

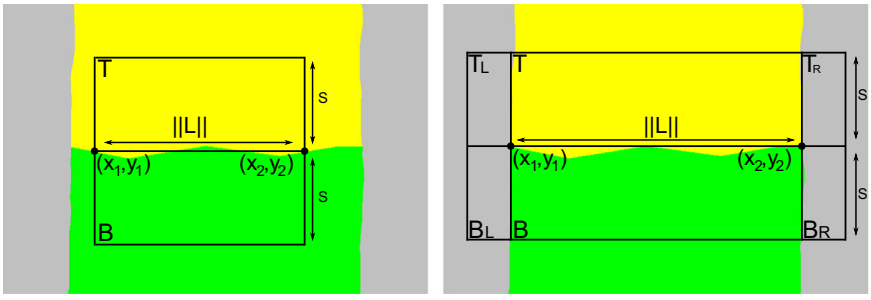

Fig. 3. Illustrations of the terminology used. The left part of the figure shows the terminology used when not considering JSD_{est} beyond the endpoints of the line; the right part illustrates all definitions and regions concerned in the calculation of $Sal(l, s)$.

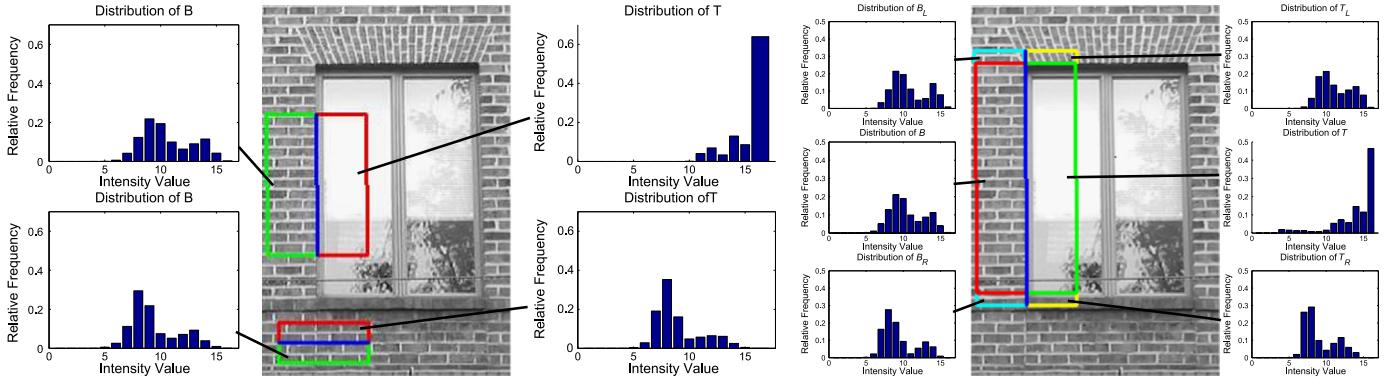

Fig. 4. Examples of pixel intensity distributions used in determining the saliency value of a line segment. Left: A comparison of distributions taken from either side of a repetitive line and from a salient line – note the similarity of distributions for the repetitive (non-salient) line. Right: A salient line segment with its endpoints localised – the distributions *beyond* the end of the segment are similar, but very different on either side of the segment.

scale s if $J(L,t) > J_{\min}\ \forall t \in [s_{\min}, s]$, where $s_{\min}$ is the minimum scale evaluated.

Thus, the algorithm proceeds by finding all lines (L,s) on an image with $Sal(L,s) > S_{thresh}$ and satisfy the three criteria given above.

3.3. Determining a representative set of lines

From the algorithm in the previous section many overlapping line segments remain (See Fig. 2(c)). We wish to cluster them and determine the most *representative* set of lines. For this, we employ the affinity propogation algorithm [24] for two main reasons: Firstly, it does not require the number of clusters to be specified beforehand. Secondly, it has been shown to be particularly effective for situations where many clusters (>50) are required – classical approaches such as k-means or Expectation–Maximisation (EM) clustering require an unfeasibly large number of restarts to obtain similar results [24].

For a given set of N lines $\mathbf{L}$, affinity propogation finds a subset $\mathbf{R} \subset \mathbf{L}$ that is representative of $\mathbf{L}$. Each line $L \in \mathbf{L}$ is mapped to its representative line by f, i.e. $f(L) \in \mathbf{R}$. Let d be a given distance measure between two line segments. Then the objective of affinity propogation is to find the mapping f that minimises the following:

$$\sum_{i=1}^{N} d(L_i, f(L_i)) \quad \text{s.t.} \quad f(L_i) = L_i \ \forall L_i \in \mathbf{R} \tag{9}$$

from which $\mathbf{R}$ may be immediately deduced. However, Eq. (9) may trivially be solved by setting f equal to the identity map and $\mathbf{R}=\mathbf{L}$, since then each of the summands is equal to zero. This is solved by letting $d(L_i, L_i) = c \forall i$, where c is a parameter of the algorithm. Eq. (9) is efficiently approximated by the max-sum algorithm (see [24] for more details).

It remains to define a distance measure d between two line segments L_i and L_j. d needs to address the *subsetting issue* correctly: if L_i is close to a subset of L_j, d should be small to reflect the large likelihood of L_i occurring if L_j does. Conversely, if L_j is close to a subset of L_i, d should be large. We use a variant of the parameter-free distance measure presented in [33]: denote the endpoints of L_i as $\mathbf{x}_1$ and $\mathbf{x}_2$ and denote the closest points on the line segment L_j to these points as $\mathbf{y}_1$ and $\mathbf{y}_2$ respectively. Then define the distance from L_j to L_i as

$$d(L_i, L_j) = \int_{t=0}^{1} \| (\mathbf{x}_1 + t(\mathbf{x}_2 - \mathbf{x}_1)) - (\mathbf{y}_1 + t(\mathbf{y}_2 - \mathbf{y}_1)) \|^2 \tag{10}$$

$$= (\mathbf{x}_1 - \mathbf{y}_1) \cdot (\mathbf{x}_2 - \mathbf{y}_2) + \frac{1}{3} \| \mathbf{x}_2 - \mathbf{x}_1 + \mathbf{y}_1 - \mathbf{y}_2 \|^2, \tag{11}$$

Eq. (11) is thus the integral of the squared distances between corresponding points on the two lines defined by $(\mathbf{x}_1, \mathbf{x}_2)$ and $(\mathbf{y}_1, \mathbf{y}_2)$.

3.4. Saliency filtering

The algorithm outlined in the previous sections exhaustively searches across all lines in an image and widths of rectangle: for an $N \times N$ image it has $O(N^5)$ complexity. Thus, an alternative approach is proposed which consists of using existing line segments determined by a fast line segment detector (e.g. [41,49]) and returning only those that are salient under our definition of line saliency. We furthermore localise the position of detected line segments under our formulation of saliency. Thus, our saliency filtering algorithm can be summarised as follows:

Inputs: Set L of line segments, parameters S_{thresh}, J_{min}, s_{min}.
Outputs: Set L' of line segments.
For each line segment $L_i \in L$:
1. Determine $s \in \{1, \ldots, \|L_i\|\}$ that maximises $Sal(L_i, s)$.
2. If $Sal(L_i, s) > S_{thresh} \wedge ((J(L_i, t) > J_{\min}) \forall t \in [s_{\min}, s])$, add L_i to L' and continue. Otherwise, go to the next line segment in L.
3. Perform hill-climbing on L_i to localise its location and width.

The hill-climbing method ensures all detected lines are *local maxima*, and five parameters of the line are altered to test for an increase in $Sal(L_i, s)$. There are two for each endpoint of the line, which are altered parallel to and perpendicular to the direction of the line segment; the other parameter is s. Each parameter is altered separately and it proceeds iteratively until a more salient position can no longer be found.

4. Depth imagery extension

The line detection algorithm described in the preceding section is not derivative-based; instead seeking informational contrast between regions from either side of a line segment. It results in a highly generalisable approach that may be applied to any situation where informational contrast can be found. Hence, it may potentially find applications in other modalities (e.g. colour or infra-red imagery). Here, we implement an algorithm for line detection in textured depth images, seeking lines that jointly delineate changes in surface orientation or texture in the same natural framework. Alternatively, if there is no texture associated with the depth imagery (as is the case for many 3D scanners), the proposed approach may detect lines that simply delineate changes in surface orientation.

For our implementation, we detect lines in a 3D structure that has been generated by multiple 'Light Detection And Ranging' (LiDAR) scans. In our case, these are coloured depth scanners that obtain the depth by measuring the time delay of a signal as it is transmitted and reflected off a 3D structure. The left of Fig. 5 shows an example of a 3D structure obtained by a LiDAR scanner. It is clear that multiple LiDAR scanners are required to recover the

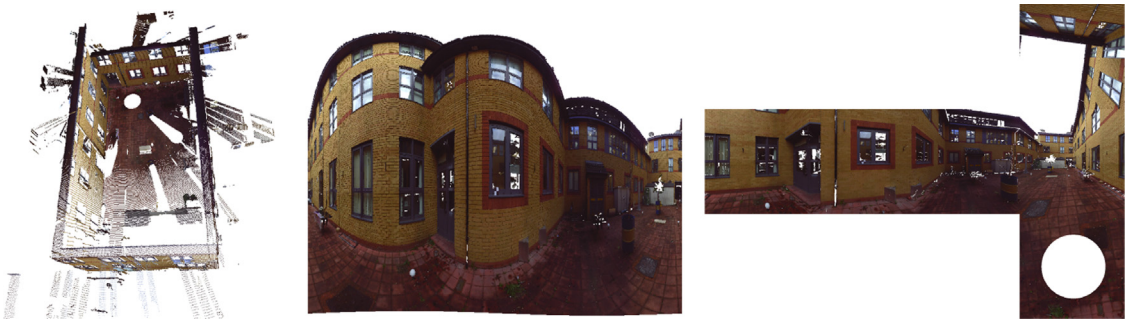

Fig. 5. Different representations of LiDAR data. Left: The 3D point cloud acquired from the LiDAR scan; Middle: A rendered spherical image taken from the location of the LiDAR scanner; Right: A rendered cubic image taken from the location of the LiDAR scanner.

structure of the scene since only points that are visible by the scanner are obtained.

It is initially tempting to detect lines directly from the LiDAR data itself. Since this is a spherical scanner, data is implicitly stored in a spherical image, similarly to the rendering in the middle of Fig. 5. However, it is evident that lines are not straight in spherical images, causing great implementation issues for our approach. Instead, the data is reprojected into a *cubic image* (right of Fig. 5) for each LiDAR scanner, with the centre of each cube at the same location as each LiDAR scanner. There is still some distortion of the lines at the edges of the cube, and to be robust to this, the cubic projection is modified slightly so each face has a field of view of 105°, providing some overlap between faces.

The implementation for LiDAR scans proceeds as follows: for each LiDAR scanner and for each face on its cubic projection, lines are detected based on both the projected texture and surface orientation. Any line that goes off the edge of the face is extended onto its neighbouring face. Subsequently, these lines are reprojected back to the 3D structure, using the approach proposed by Buch et al. [13]. Finally, 3D line segments are combined from multiple LiDAR scans using a similar affinity propogation approach as outlined in Section 3.3.

4.1. Line detection in textured depth imagery

For each face of the cube the algorithm proceeds in the same way as in Sections 3.1–3.3 except for the representation of the distributions $\mathbf{m}$ and $\mathbf{n}$. Since our aim is to detect lines that jointly delineate changes in texture or surface orientation (or just surface orientation, if there is no texture data available), they need to represent both the *direction of the normals* and optionally the intensity of the projected depth image. The normals are estimated from the depth data by a least squares plane-fitting approach from a small neighbourhood about each point.

In constructing $\mathbf{m}$ and $\mathbf{n}$, b_i and b_n bins are used to represent the intensity and direction of the normals respectively, with an extra bin when there is no data present, resulting in $b_i b_n + 1$ dimensional histograms. The b_i intensity bins are the same as in the 2D implementation, while the normals are binned uniformly across the surface of the sphere. The latter is a challenging problem for general b_n, so it is restricted to determining which vertex of a given *Platonic solid* it is closest to. We use the regular icosahedron ($b_n = 12$), however $b_n = 8$ and $b_n = 20$ also gave good initial results. In the case where there is no intensity data present, lines are detected based purely on the direction of the normals, resulting in a $b_n + 1$ dimensional histogram.

From these constructions, lines are detected in the same way as for the 2D implementation. However, if a resulting line may be extended by a small amount such that it is partly off the image, it is considered as being part of two faces. In this case, its endpoint is extended along the neighbouring face and its saliency value is computed here in pixel intervals. The new (cubic) position of the line is deemed to be where this attains its maximum value. Note that the area either side of the line is still well-defined in this case (as the union of areas on each face) meaning its saliency and its reprojection (in the next section) operate in the same manner.

4.2. Line reprojection

Line reprojection is required in order to convert lines detected in the previous subsection into 3D line segments. This is not as trivial as simply reprojecting the endpoints back since it may cause large errors when the endpoints are slightly misaligned and fall on different planes, or fail completely when there is no depth data available at one point. Therefore, we use the approach proposed by Buch et al. [13] which, for completeness, is briefly outlined here. They propose to reproject lines according to the *type* of line it is – whether the line is caused solely by a change in image intensity, a change in orientation of the normals, or a change in depth. Fig. 6 shows these three cases. Each case relies on locally approximating two planes (P_1, P_2) from rectangular regions either side of the line, or a plane P_{all} from a rectangular region surrounding the line, each by a RANSAC approach to plane estimation. The back-projected plane of the 2D line needs to be considered here, which will be denoted by P_L.

If the distance between the centroids of the points in P_1 and P_2 is large, it is likely that the line is caused by a depth discontinuity – in this case, the reprojected line is the intersection of P_L and the closest plane to the camera between P_1 and P_2. If the angle between the normals of P_1 and P_2 is larger than a given threshold, then the line is due to an orientation discontinuity. Here, P_L is intersected with both P_1 and P_2 and the mean is selected as the reprojected line. Alternatively if the angle is sufficiently small the line is due to a change in image intensities – the reprojected line is thus the intersection of P_L and P_{all}.

4.3. Line clustering in 3D

In this stage, reprojected lines from multiple LiDAR scans are combined and clustered. This may be done using affinity propogation as previously defined – note that the distance between line segments (10) is well-defined in any dimension. However, from multiple LiDAR scans, some reprojected lines are more accurately located than others (due to the relative positions between the lines and the scanners). Hence, the distance $d(L_i, L_j)$ (10) is redefined as $\tilde{d}(L_i, L_j) = d(L_i, L_j)/A(L_j)$, where $A(L_j)$ denotes the *accuracy* of line L_j, in order to favour more accurate line segments.

To compute the accuracy, first define the vector from the camera centre to the midpoint of L_j as $\mathbf{v}$. Let $\mathbf{n}$ denote the normal to the plane that L_j is on (If L_j lies on the intersection of two planes, compute the accuracy with respect to each plane and take the average). Denote the angle between $\mathbf{v}$ and $\mathbf{n}$ as θ and denote the field of view *per pixel* as ϕ. Then the 3D distance subtended by one pixel is given by

$$d_P = \frac{\|\mathbf{v}\| \sin(\phi)}{\cos(\theta + \phi)} \tag{12}$$

Subsequently the accuracy is defined as $A = 1/d_P^2$, measuring how many (square) pixels subtend a square metre from the image.

5. Experiments

In this section we evaluate the performance of our proposed approaches against other line detectors. We compare against the Progressive Probabilistic Hough Transform (*PPHT*) [41] – a classical method for line detection, and the state-of-the-art *LSD* algorithm

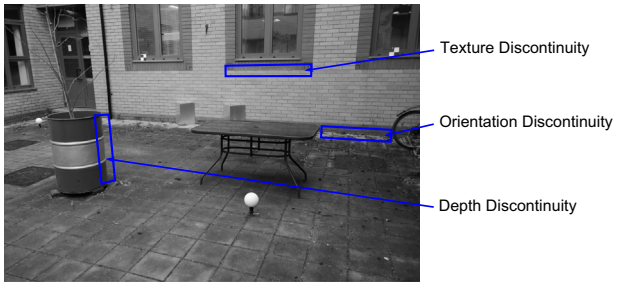

Fig. 6. Illustration of the different types of lines from a textured depth image: The top line is caused by a change in texture on the surface of the wall; the middle line by a discontinuity in the orientation of the surface normals; the bottom by a depth discontinuity.

[49] by Grompone von Gioi et al. Three variants of our approach are used: the full saliency detector *Sal*; a pure filtering approach applied to *LSD* lines referred to as *LSDF*; and a filtering approach with subsequent localisation using our saliency measure, referred to as *LSDF-Loc*. We start by giving implementation details of our approaches in Section 5.1 and describe the evaluation measures used (*repeatability* and *registration accuracy*) in Section 5.2. Subsequently results are presented, in Section 5.3 for 2D line detection and in Section 5.4 for 3D line detection.

5.1. Implementation details

It was stated in the methodology section that the algorithm *Sal* goes through each possible line segment (L, s), determining its saliency value and accepting if it is above a given threshold and satisfies a number of other conditions. Affinity propogation is subsequently used to determine the most representative set of lines.

In the first instance, all line segments are considered by evaluating the saliency measure across all horizontal lines, then repeating this process r times on evenly spaced rotations of the image, where we take $r=45$. To do so requires $JSD_{est}(\mathbf{n}, \mathbf{m})$ to be determined from a set of pixels. Here the pixel intensities are bilinearly interpolated into 16 bins. The line segments beyond the end of the line (L_L and L_R) are of a fixed length of 6 pixels. We use $S_{thresh} = 0.3$, $J_{\min} = 0.15$ and $s_{\min} = 2$. In the affinity propogation stage, we have found the parameter $d(L_i, L_i) = 700$ to be effective.

Since it is a particularly slow algorithm ($O(N^5)$ for an $N \times N$ image) the image is initially downsampled to a width of 200 pixels, and detected lines are subsequently refined using the algorithm outlined in Section 3.4 at its true size (in a coarse-to-fine approach).

5.2. Evaluation measures

In this subsection the terms *repeatability* and *registration accuracy* are defined. They are both measures that are defined between sets of line segments detected on a pair of images under a known homography.

5.2.1. Repeatability

For a pair of images with known homography relating them, the *repeatability* for a set of line segments detected on each image by a given detector is computed as follows: first, the known homography is applied to one of the sets of lines. Define the distance between two lines as the minimum Euclidean distance between the lines' endpoints. Then, for each line projected under the known homography, its nearest neighbour (NN) is computed in the other set. If the distance between the two lines is less than a given threshold, this is deemed a correspondence. Then the *repeatability* is the number of correspondences divided by the minimum of the number of lines in each set.

However, Hauagge and Snavely [27] note that this measure is biased towards detectors that produce a lot of features, and a measure that is invariant to the number of lines detected is proposed. We proceed as follows: for each set of lines on an image, order them in decreasing value of saliency. For *LSD*, the lines are ordered in decreasing order of another response value – the probability of detection in random noise. For *PPHT* the lines are simply ordered by length. Then, for given natural numbers k up to a specified limit (we take 150 here), take the first k lines in each set. The repeatability of these k lines is subsequently calculated, and a graph can be plotted of repeatability against the k most responsive lines in each set (here, the repeatability is determined within a distance threshold (t) of 5, 10, 15, and 20 pixels).

5.2.2. Registration accuracy

Here, a pair of images are registered by computing the homography between them. The *registration accuracy* gives an indication of the similarity between this and the ground truth homography. Again, we perform this in a way that is invariant to the number of lines detected, plotting the proportion of homographies recovered within a threshold against the most responsive k lines. To compute a homography, we implement the MSLD [51] descriptor for line segments, allowing us to determine putative correspondences between line segments in different images by the similarity of their descriptor. The homography is subsequently recovered using the Direct Linear Transform (DLT) with small sets of *corresponding endpoints*, and using RANSAC to determine the homography with the largest number of inliers.

The homography could have instead been calculated using line correspondences, where a line is defined to be infinitely long and information about its endpoints is discarded [52]. However, we observed the results were poorer for this approach than point-based homography estimation with line endpoints – this could be for two reasons. Firstly, infinitely long lines discard valuable information and are redundant in cases where a continuous line is segmented in many places (as is often the case in urban scenes). Secondly, we have good reason to assume the endpoints of the lines are matched up reasonably accurately since the MSLD descriptor has already matched line segments – this would not be the case if the endpoints were not sufficiently aligned.

To determine the *registration accuracy* we aim to give a measure of how accurate the recovered homography is against the ground truth homography. To do so, one might decompose the homography into rotation and translation parameters and compare their errors, however this can only be done if the intrinsic parameters are known (which they are not). We therefore resort to other measures. Our measure of goodness-of-homography-estimation is as follows: take a pixel on the first image and apply the known homography and estimated homography to it and find the squared distance between these two projected points. Take the average of this over all pixels in the image. Then, do the same, but in the other direction (i.e. with the inverse homography), and square root the final result (to give an RMS error). Thus our measure is an approximation to the following:

$$d(G,H) = \sqrt{\frac{1}{XY}\int_{y=0}^{Y}\int_{x=0}^{X} \|H(x,y) - G(x,y)\|^2 + \|H^{-1}(x,y) - G^{-1}(x,y)\|^2 \, dx\, dy} \tag{13}$$

where G and H are homography transformations and X and Y are the number of rows and columns respectively in the pair of images. We are unable to find a closed form solution to Eq. (13) (note that $H(x,y)$, $G(x,y)$ are non-linear since computations are done via projective space), hence we resort to the approximation outlined in the above paragraph. *Finally*, so as to be robust to outlying homography estimates, we determine the proportion of homography estimates such that $(d(G,H) < t)$ where t is equal to 5, 10, 15, and 20 pixels.

5.3. 2D line detection results

In this section, both qualitative and quantitative results are presented across a range of imagery, with qualitative results presented in Section 5.3.1. For a quantitative evaluation, the performance of the line detectors is tested on a set of images of building facades from [16] (Section 5.3.2); their robustness to Gaussian noise from the same set of images (Section 5.3.3), and their robustness to a range of image transformations from the dataset presented in [53] (Section 5.3.4). Finally the performance of existing line detectors at different scales is tested (Section 5.3.5).

The *repeatability* and *registration accuracy* is determined between pairs of images under their known homography (which has been calculated manually for the building facade dataset [16], but is known from the dataset presented in [53]).

5.3.1. Qualitative results

Qualitative results for 2D line detection are shown[3] in Fig. 7. It is noticeable that *Sal* naturally avoids repetitive areas in the brick facades for the top two images, and detects the geometric structure of the scene in the third image. In the fourth and fifth images, *Sal* further avoids repetitive areas in the scene, while *LSDF* and *LSDF-Loc* avoid them to a lesser extent. The sixth and seventh images show the effects of compression and occlusion on line detection respectively [53], where it can be seen that *Sal* detects the broad underlying structure of the scene. This implies our approach has potential applications for compression tasks, further demonstrated by quantitative results in Section 5.3.4. The bottom two images are of building facades from the experiments presented in Section 5.3.2.

Across the range of images, *PPHT* detects many erroneous lines, largely due to the fact that it does not take into account the direction of the gradient of pixels in its lines. *LSD* detects all line segments on the image based purely on the local image derivative, whereas *Sal* tends to detect the structurally important lines. *LSDF* and *LSDF-Loc* avoid some of the repetitive areas and cull many non-salient lines detected by *LSD*.

5.3.2. Quantitative evaluation on building facades

In this section, the performance of the line detectors is tested on a set of 12 image pairs of building facades taken from the dataset presented in [16], see the top of Fig. 9 for examples of the dataset and Fig. 7 for some qualitative results. The average number of line segments detected per image for this dataset is as follows: *PPHT* - 634.9, *LSD* - 1738.7, *Sal* - 274.3, *LSDF* and *LSDF-Loc* - 1137.6; with the average execution times: *PPHT* - 0.167 s, *LSD* - 0.182 s, *Sal* - 325.96 s, *LSDF* - 6.05 s and *LSDF-Loc* - 14.02 s. The detection of a large number of lines is potentially problematic in a registration context since it can lead to fragmentation of prominent lines; the detection of many similar, repetitive lines that are difficult to match; or a *significantly* slower registration process if correspondences between lines also need to be established. Therefore, qualitative results for the top-50 lines are shown in Fig. 8 to take account of the number of lines per detector. Here it can be seen that, while *PPHT* and *LSD* detect the longer lines, there is more repetition in their detections: *Sal* on the other hand provides a more complete description given the same number of lines.

The quantitative results are shown in Fig. 9. The left-most graph simply shows repeatability against threshold, without taking into account the number of features produced by each detector. *LSD* performs the best, with *LSDF* and *LSDF-Loc* performing similarly for smaller thresholds, but slightly worse for larger thresholds. The repeatability results for various thresholds are shown on the first row of Fig. 9, where it can be seen that *LSDF-Loc* performs the best, with *LSDF* close behind. For $k < 100$, *Sal* performs better than *LSD*. The second row shows results for registration accuracy, where similar conclusions can be drawn: regardless of the threshold used, all three of our proposed methods (*LSDF*, *LSDF-Loc* and *Sal*) perform better than other methods, while *PPHT* consistently performs poorly.

[3] Second image by Colin Smith, http://commons.wikimedia.org/wiki/File:Georgian_House,_Farnham_-_geograph.org.uk_-_1622126.jpg. Third image by Tony Atkin, http://commons.wikimedia.org/wiki/File:Buckland_Monachorum_Church_-_geograph.org.uk_-_803201.jpg. Both images licensed under CC BY-SA 2.0 and are grayscales of original.

5.3.3. Robustness to noise

Here the performance of the line detectors in the presence of Gaussian noise is tested. The same dataset of building facades as used in the previous section are used here, with varying levels of Gaussian noise added to each image. The top section of Fig. 10 shows qualitative results of line detection in increasing noise. With the exception of *PPHT*, all methods detect fewer lines in more noisy images.

Again, the repeatability and registration accuracy are measured for increasing levels of noise. In the first case, the repeatability of the top k lines of the line detectors is measured for a threshold t of 10 and 200, and where k is equal to 50 and 100 (thus producing four graphs), see the top four graphs in Fig. 10. For smaller levels of noise, *LSDF-Loc* performs best, with *Sal* performing better for higher levels: *Sal* records very little drop in performance in increasing noise.

In the second case, the proportion of homography estimates less than a threshold t are measured in increasing noise. Again, four graphs are produced, by varying t and k in the same manner. It is observed in the bottom four graphs of Fig. 10 that *Sal* and *LSDF-Loc* outperform the other methods, with *Sal* performing better when only the top 50 lines are used rather than 100. This shows the strength of salient line segment detection – its ability to detect segments indicative of the underlying geometry of the scene, unaffected by local perturbations of the image.

5.3.4. Robustness to image transformations

In this section, the performance of the line detectors is tested across a range of image transformations, according to the dataset by Zhang and Koch [53]. They include eight groups of transformations with six images in each group, with a known underlying homography between images for each group. Three of the groups are taken from [42]. Two example images from each group are shown at the top of Fig. 11 with the results at the bottom. Qualitative results for a compressed image and an occluded image from the dataset are shown in Fig. 7 – it is observed that *Sal* more easily detects the salient line segments than other approaches, and explains its strong quantitative results (Fig. 11).

We solely test the repeatability for the top-50 and top-100 lines here. It is observed that our approaches consistently outperform *PPHT* and *LSD*. The only exceptions are in low texture and with scale changes, where they obtain a similar performance. Particularly for low texture this is not surprising – our approach is beneficial due to its ability to naturally avoid textured areas, clearly giving no benefit for low textured scenes. *Sal* performs particularly well for both compression and blurring – transformations that remove fine details but preserve the broad structure of the scene; consistent with the idea that it detects the *salient* aspects of the image. Again, *LSDF-Loc* often outperforms *LSDF*, however it will never perform as well as *Sal* for some transformations (e.g. compression) where the initial set of lines obtained by *LSD* are poor. Furthermore, the results demonstrated here are, overall, better than those given in the previous section, where *Sal* obtained a similar performance to *LSD* with the top-100 lines selected.

5.3.5. Scale variant evaluation

In this section we compare the existing state-of-the-art line detector, *LSD*, at different scales and compare to our proposed approach *Sal*. To do so, an image is downscaled by a given percentage and the *LSD* algorithm is run on the downscaled image. Results are shown in Fig. 12, where *LSD* is tested on downscaled images of 25%, 50%, 75%, and 100% (i.e. full resolution). The quantitative results are performed on the building facade dataset presented in [16]: exactly the same quantitative evaluation is performed as in Section 5.3.2. The qualitative results show that, at the higher scales, *LSD* detects fewer lines in repetitive structures (particularly evident in the first image of Fig. 12). This is to be

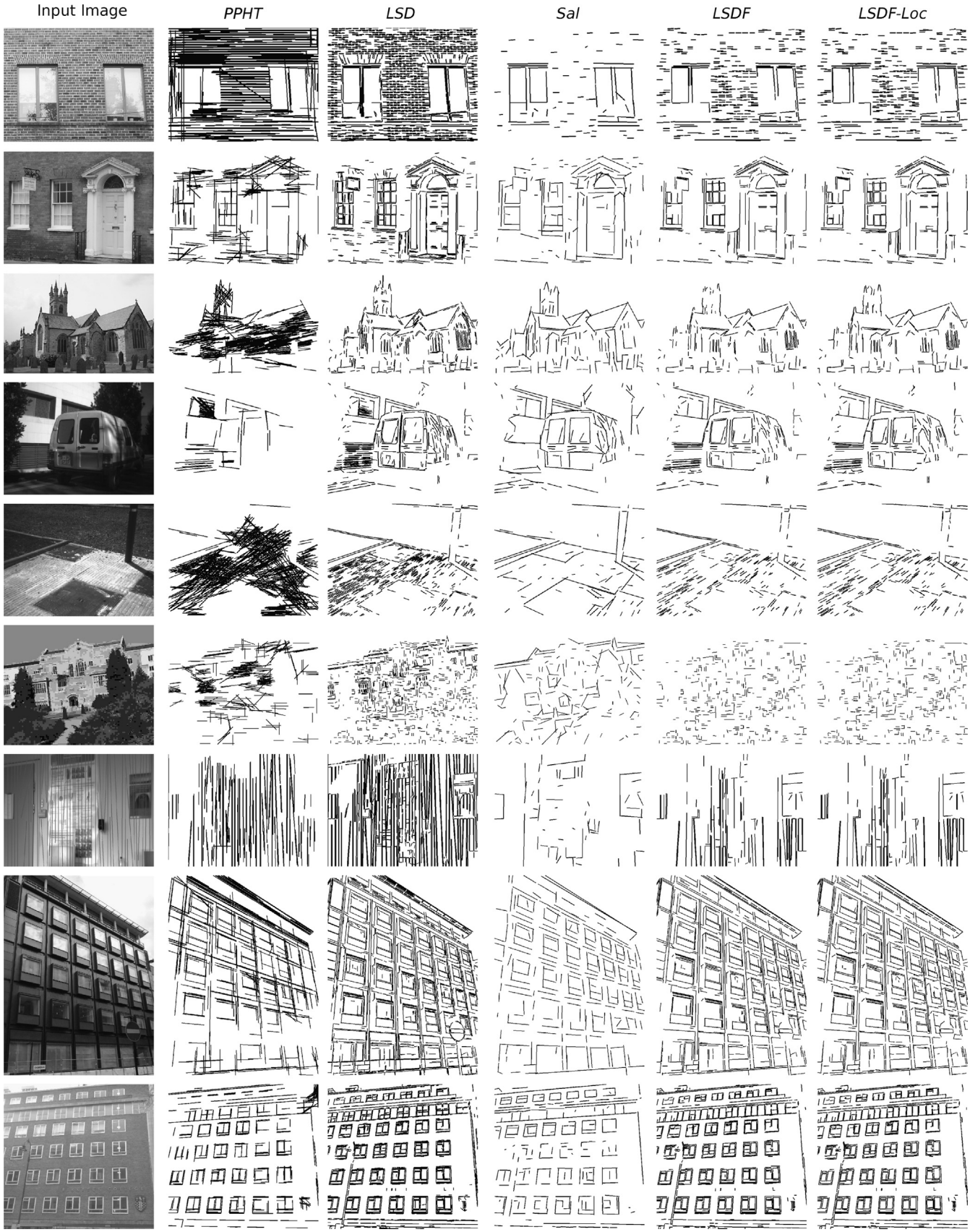

Fig. 7. Qualitative results of line detection on a range of images. From left to right: Input image, *PPHT*, *LSD*, *Sal*, *LSDF*, *LSDF-Loc*. Fourth and fifth images are from [7,2]; sixth and seventh from [53]; eighth and ninth from [16].

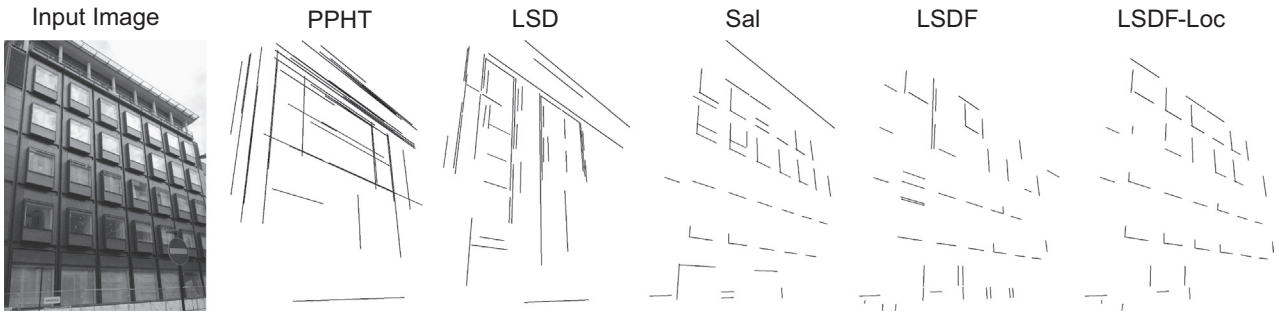

Fig. 8. The top 50 lines according to each line detector.

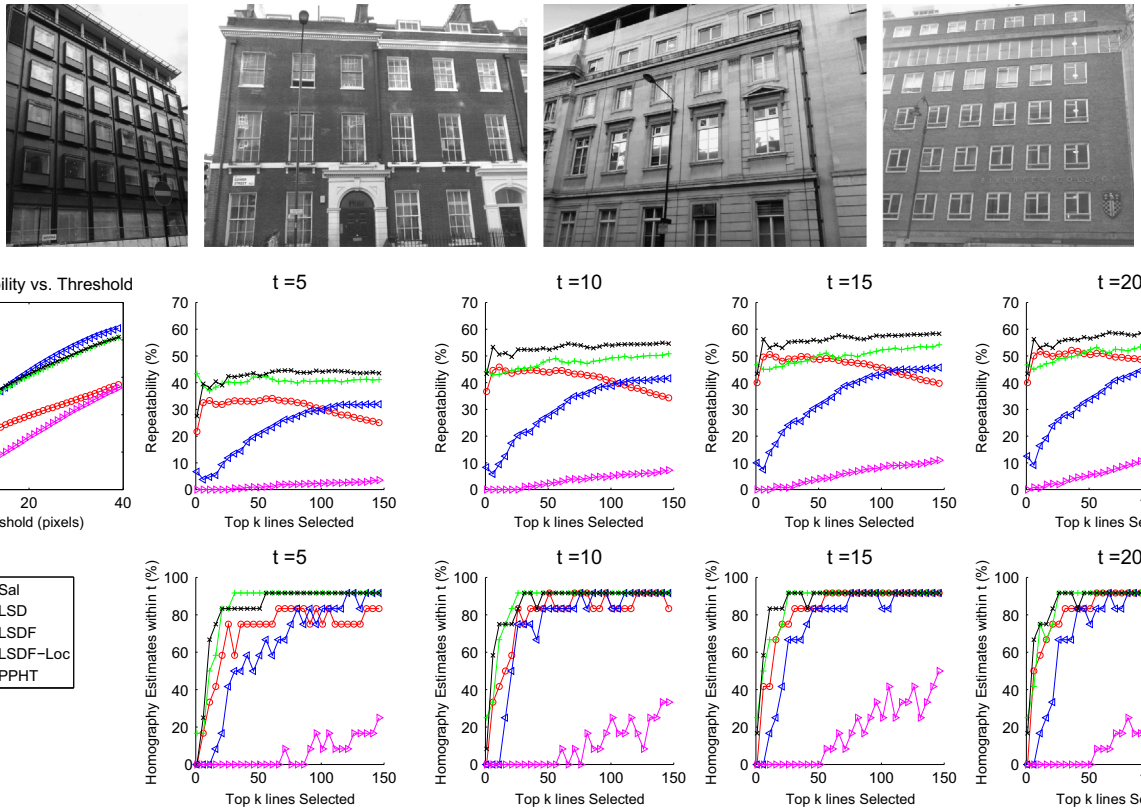

Fig. 9. Quantitative results of line detection on building facades. Top: Example images from the dataset. Bottom: Left: Repeatability vs threshold; Top: Repeatability vs top k lines selected for varying thresholds; Bottom: Registration accuracy vs top k lines selected for varying thresholds.

expected, as downscaling typically results in an image without fine detail. It is further demonstrated on the quantitative results where *LSD* at 75% and 50% perform slightly better than *LSD* at 100% (but *Sal* still performs significantly better). However, *LSD* is scale-variant, and it is difficult to know a priori the optimal scale. The results suggest that a downscaling to 25% is too high a scale to be of use, yet *LSD* at 50% still detects some fine-detail structures (e.g. the windows of the church). Our approach, *Sal*, is scale-invariant, allowing it to naturally avoid repetitive structures while detecting lines of variable widths. Furthermore its generalisable formulation, dependent on image statistics rather than image gradient, allow it to be naturally extended to depth imagery.

5.4. 3D line detection results

In this section we evaluate our method for line detection for LiDAR data as described in Section 4. The parameters used are the same as for the 2D saliency detector, with the exception of the prior α and the parameter $d(L_iL_i)$ used in the affinity propogation stage. In the first case α is decreased to 0.25 because the distributions are split into many more bins and $\alpha = 1$ is noticed to favour uniform distributions too strongly for such a large number of bins. For the second case, $d(L_i, L_i)$ is in proportion to the size of the model: 0.002 times the diameter of the bounding box of the model is used for this. Whilst the approach described in Section 4 describes line detection from an {intensity + depth} image, it can be just as easily implemented by reprojecting lines using just the intensity or just the depth data separately. We shall refer to results from these three cases as *both*, *intensity*, and *depth* respectively. There are four datasets used: three of them from [35] (*Courtyard*, *Plaza*, *Reception*)[4] and one from the SCENE project [1] (*Room*), all of which are shown in Fig. 13. They have been generated from multiple LiDAR scans, with the largest, *Plaza*, generated from seven scans.

[4] *Courtyard* refers to the Outdoor capture in Section 2 of the dataset presented in [35]. *Plaza* and *Reception* are both in Section 5 of the dataset of [35].

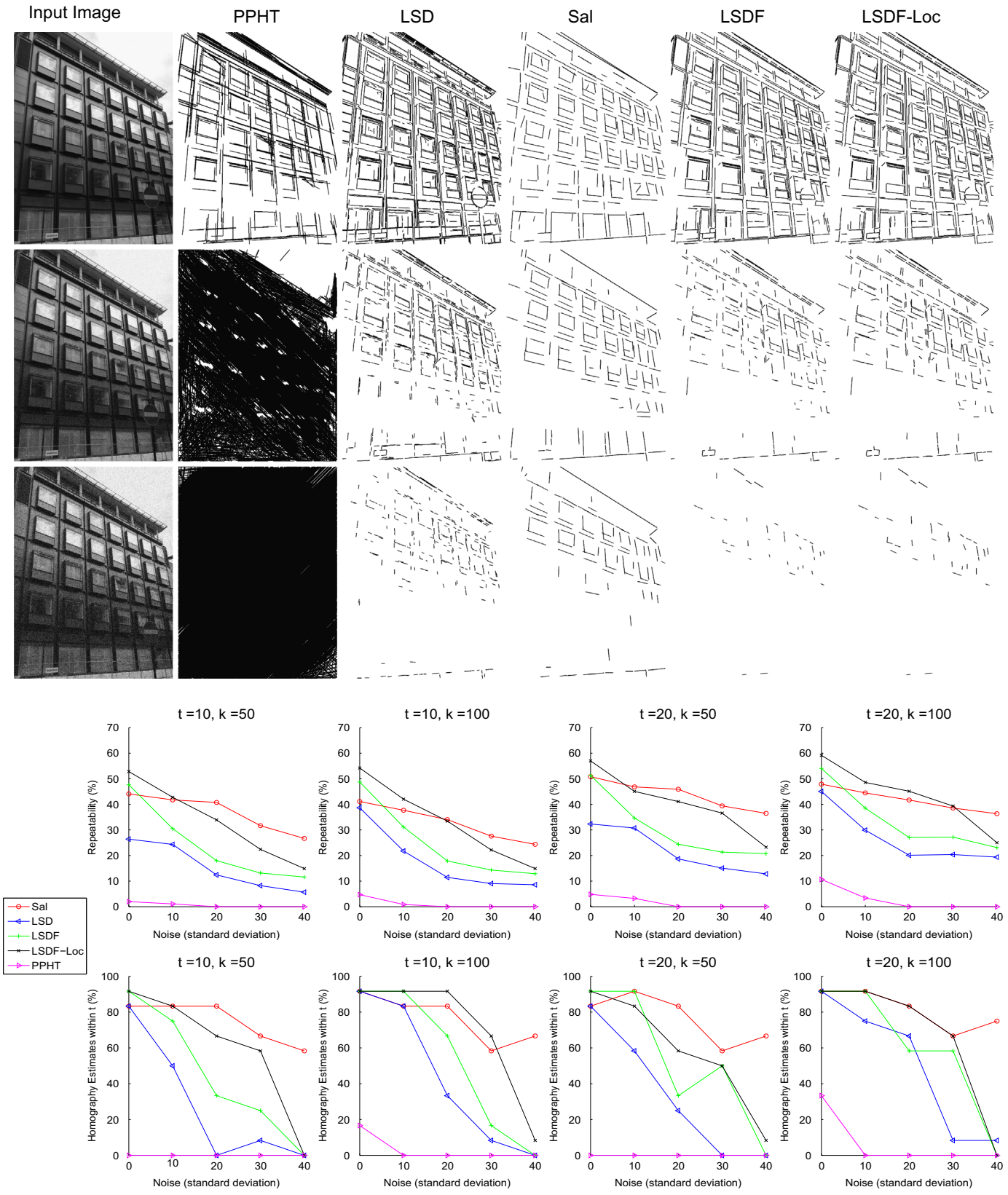

Fig. 10. Top: Qualitative results. From left to right: Input image, *PPHT*, *LSD*, *Sal*, *LSDF*, *LSDF-Loc*. From top to bottom: zero noise, noise (s.d. 20), noise (s.d. 40). Bottom: Quantitative results of line detection in noise. Top row: Repeatability against noise; Bottom row: Homography estimation within t against noise. Results are plotted for ($t=10$, $k=50$), ($t=10$, $k=100$), ($t=20$, $k=50$), and ($t=20$, $k=100$).

5.4.1. *Qualitative evaluation*

Qualitative results for these are given in Fig. 14 for lines detected using just the *depth* component, just the *intensity* component, and *both* components. A number of observations can be made here. Firstly, there is always a circle detected on the ground beneath each LiDAR scan – this is unavoidable since the scanner does not have

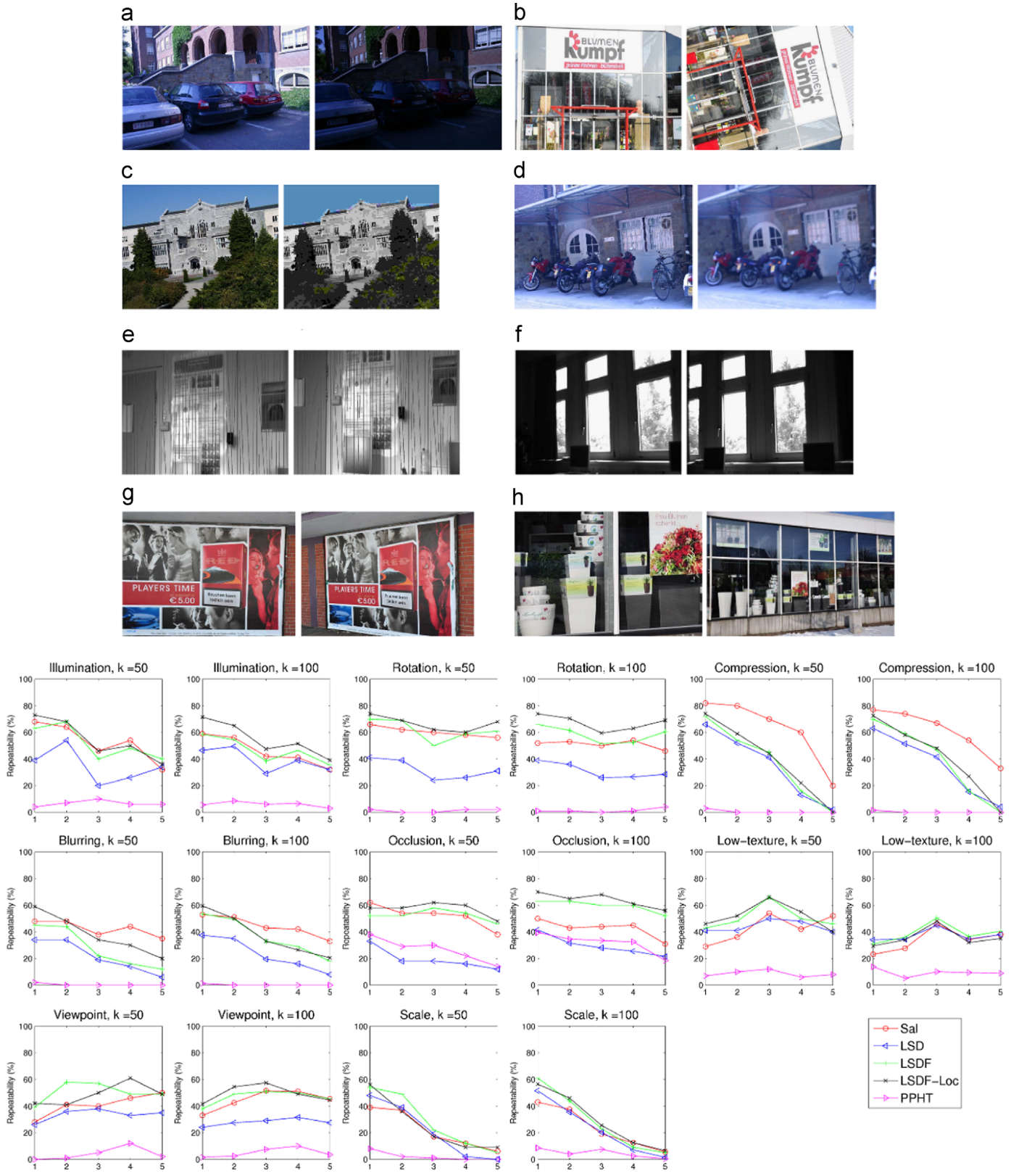

Fig. 11. Top: Examples in the image dataset for eight groups of image transformations. For each group, there are six images in the dataset ranging from small to large transformations, with the first and last images in each group shown here. Bottom: Results for each group of image transformations. The repeatability is measured, taking the top 50 and 100 lines in each case (a) illumination (b) Rotation (c) Compression (d) Blurring (e) Occlusion (f) Low-texture (g) Viewpoint and (h) Scale.

complete spherical vision, and the same happens when lines from other line detectors are reprojected to 3D (see Fig. 15). Secondly there is, for the most part, a reasonably high overlap between lines from *intensity* and lines from *depth* – typically due to depth discontinuities in the data. This may be observed particularly on the windows of the *Courtyard* dataset where there is no data

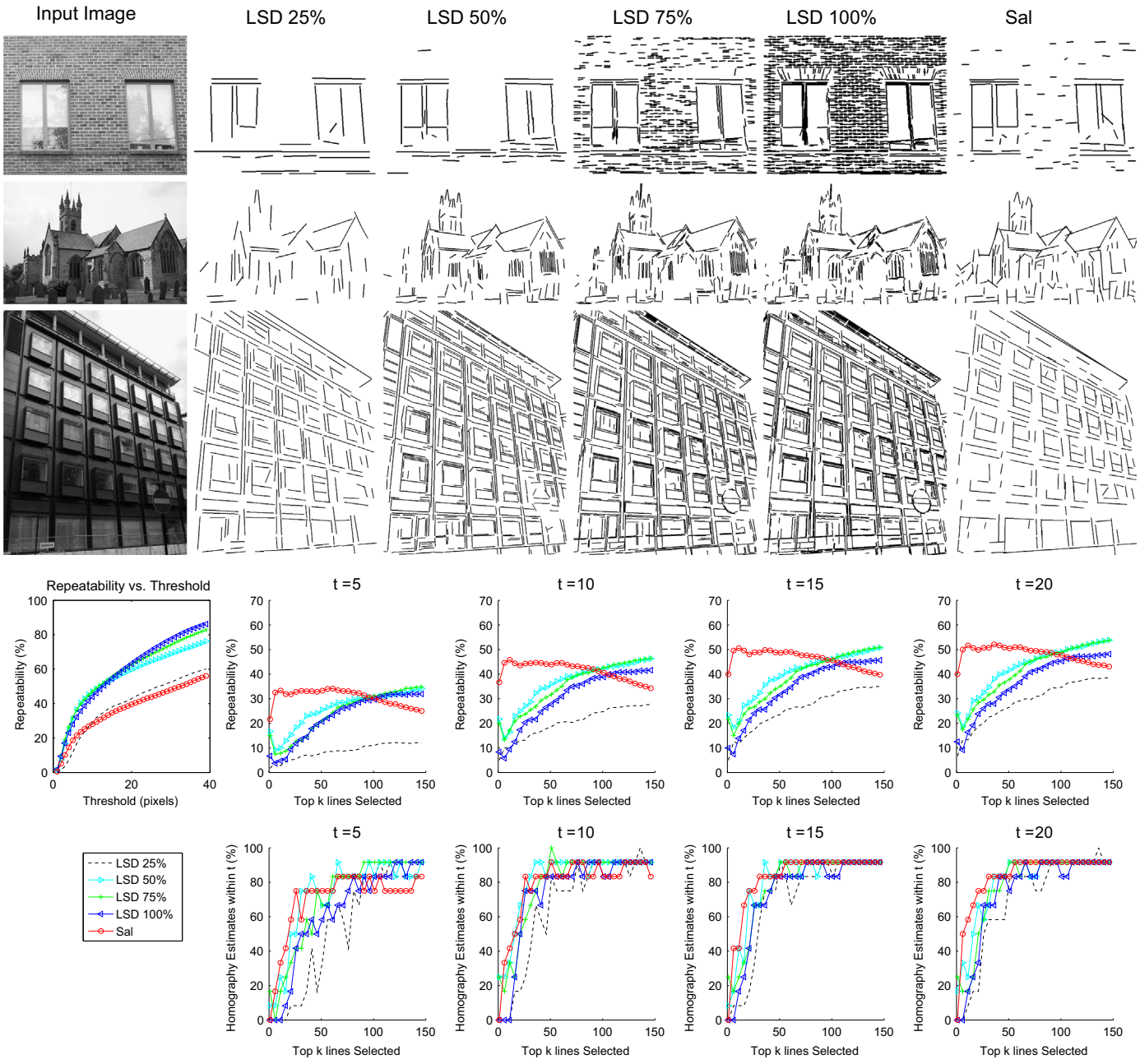

Fig. 12. Top: Qualitative results for *LSD* at different scales, compared to *Sal*. Bottom: Quantitative results from the dataset presented in [16], showing repeatability vs threshold (Left); Repeatability vs top k lines selected for varying thresholds (Top); Registration accuracy vs top k lines selected for varying thresholds (Bottom).

Fig. 13. LiDAR Datasets. From left to right: *Courtyard*, *Plaza*, *Reception*, *Room*.

present on one side of the line, hence deemed salient by both the *intensity* and *depth* approach. However, there are some important differences, e.g. the edges of *Room* and *Courtyard* are crisply detected by the *depth* approach and avoided by *intensity*. When considering *both*, both geometric and textural lines are detected within the same framework (particularly evident in *Room*).

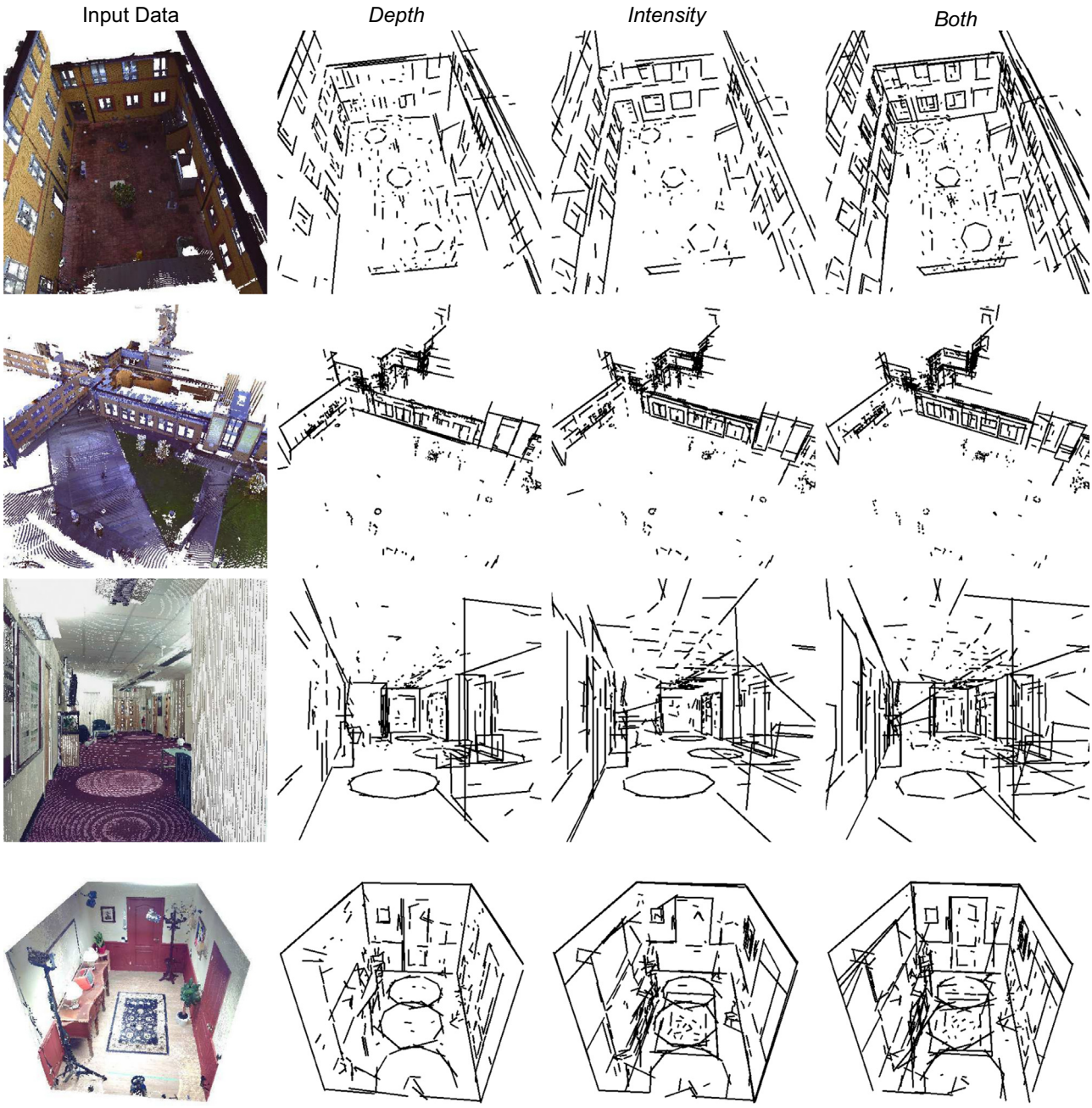

Fig. 14. Lines detected in each LiDAR dataset when considering just the *depth* component, just the *intensity* component, and *both* components respectively.

5.4.2. Quantitative evaluation

Here, we give multi-modal results for when just the intensity component and just the depth component are considered. Fig. 16 gives an example of such images: the depth component is rendered in such a way that the colour represents the direction of the normal.

Any other 2D line detector, as used in the previous section, may be used to detect lines on each face of a cubic image when just the intensity *or* depth component is considered. Hence, for a single LiDAR scan, we may consider only one of the components and detect 2D lines using any other approach (e.g. *PPHT*, *LSD*) on each face of its cubic image and backproject to 3D. However, using other approaches, lines should not be combined from multiple LiDAR scans using affinity propogation; this is designed to find a *representative set of clusters*, rather than to cull a small number of repeated segments from multiple views. Hence, for a fair qualitative comparison, we compare reprojected line segments taken from one component of just a single LiDAR scan.

Qualitative results from four LiDAR scans (one from each dataset) are shown in Fig. 15. It can be observed that, similarly to results in 2D (Fig. 7), *Sal* naturally avoids repetitive parts of the scene where others do not, particularly for the brickwork near the LiDAR scanner in the *Courtyard* dataset, and the tiled ceiling in *Reception*. The reprojection to 3D further demonstrates the ability of our approach to detect lines that are representative of the underlying aspects of the scene. This results in an often greater similarity between *intensity* and *depth* for *Sal* than there is for other methods, further demonstrating its applicability for multi-modal data.

Now quantitative results are discussed, between lines detected solely from *intensity* and solely from *depth* for a 2D image across a

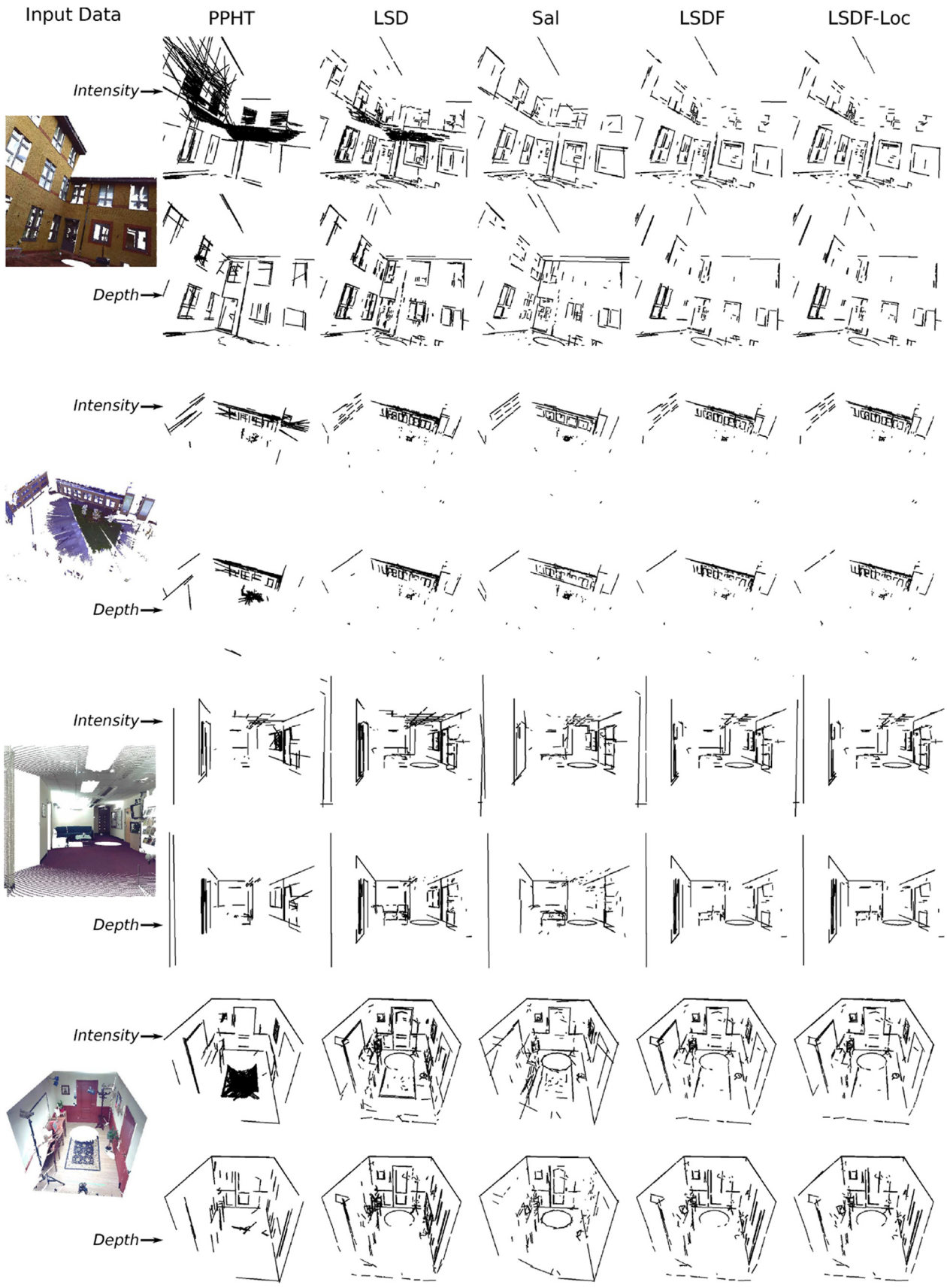

Fig. 15. Lines detected by each 2D line detector using just the *intensity* component in a cubic image of a single LiDAR scan, backprojected to 3D.

change in viewpoint. The images are rendered from the point cloud of a given LiDAR scan, with the camera located at the same location of the LiDAR scan, and lines are detected based on the *intensity* component of the image. The repeatability between these lines and 3D lines that have been detected by *another* LiDAR scan using the *depth* component is then measured, and results are

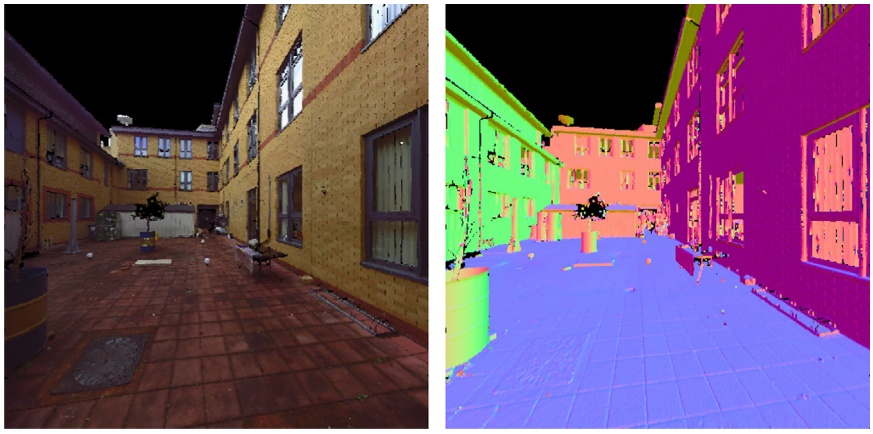

Fig. 16. Left: Intensity rendering, Right: Depth rendering (the colour represents the direction of the normal at that point).

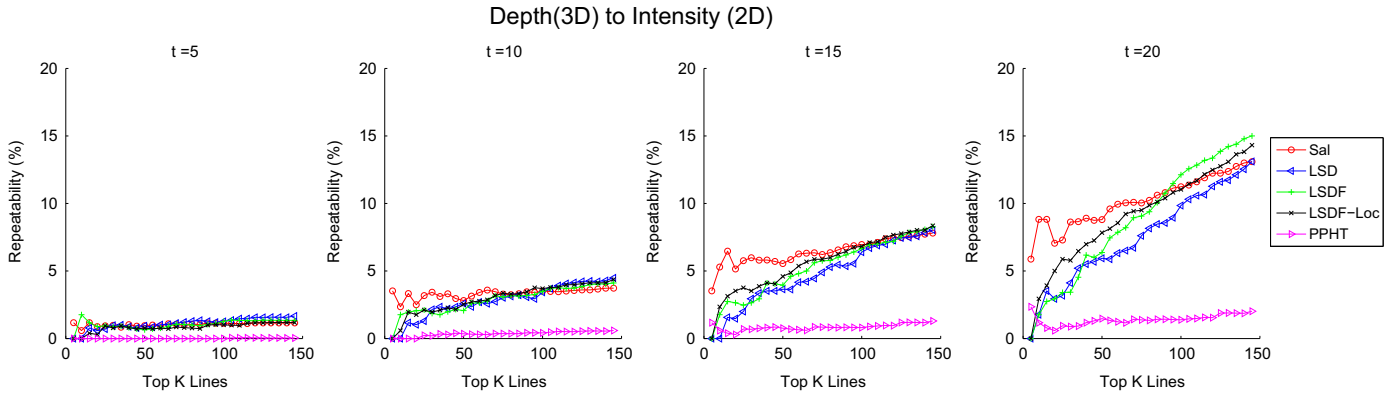

Fig. 17. Repeatability of the different line detectors. Top row: Repeatability between reprojected line segments using just the *intensity* component of a single LiDAR scan (3D) against segments detected by just the *depth* component of a 2D image, for a range of thresholds. Bottom row: Repeatability between 3D segments using just the *depth* component against the 2D segments using just the *intensity* component.

shown in Fig. 17. They indicate that *Sal* performs the best, particularly so for smaller numbers of lines. It demonstrates that *Sal* detects lines that are often *geometrically salient* and are potentially applicable for multi-modal registration (e.g. for the case of registering an image to an untextured LiDAR scan). Furthermore, we wish to emphasise that the results here are using *Sal* only for the sake of comparison (by constraining it to only the *depth* component), and that it has the other qualitative advantages of being able to detect both *textural* and *geometric* lines simultaneously, as well as naturally combining line segments between LiDAR scans.

6. Conclusions and future work

In this paper we have presented a novel, distribution-based approach to line detection. Whereas other line detectors simply detect lines based on the image gradient, our approach explicitly takes into account the surroundings of a line, resulting in a line segment detector that naturally avoids repetitive areas and returns lines that are *representative* of the structure of the scene. Furthermore, its highly generalisable formulation makes it readily applicable to other modalities, as demonstrated by an extension to depth imagery, where lines that jointly delineate changes in surface orientation or texture are detected. For fast salient line segment detection, a filtering approach is proposed, often yielding similar results as the full saliency approach. The results indicate that our approaches achieve superior repeatability across a range of transformations compared to other line detectors and the multi-modal results indicate that they naturally detects lines representative of the structure of the underlying scene. Not only is it of potential use in registration contexts as evaluated here, but also for compression related tasks as demonstrated by its high repeatability under this transformation.

There are potential areas for further improvements – in particular, the good results obtained by filtering methods (*LSDF* and *LSDF-Loc*) indicate that an approach that combines local *and* regional information about a line has potential benefits. Such a local and regional approach would have similarities with approaches to the more general problem of saliency detection in images. However since our approach is, to the best of our knowledge, the first distribution-based approach to line detection, we consider such a two-tier system beyond the scope of this research.

Future work will include the registration of lines between different modalities (e.g. 2D and 3D). For this problem, the correspondences between line segments need to be determined – thus it is referred to as the *Simultaneous Pose and Correspondence* (SPC) problem. It is a computationally expensive problem [20] (for N 2D lines and M 3D lines, it has complexity $O(M^2N)$) so any method that has a high repeatability for a smaller number of lines will be far more suited to this kind of problem. Hence it is anticipated that the approach proposed here will be of great use

for the more general problem of pose estimation, not only for its ability to detect the structure of a scene in a small number of lines, but also its unified approach to line detection in multi-modal data.

Research data

To facilitate repeatable research, all data used here that is not currently available as research data is now made available. Details are available for the Room dataset at [36]; for the Courtyard, Plaza and Reception datasets at [34]; and at [12] for images used that are not part of any cited dataset.

Conflict of interest

None declared.

Acknowledgements

This work was supported by the EPSRC (grant number EP/K503186/1) and the EU ICT FP7 project IMPART (grant number 316564).

References

[1] Novel scene representations for richer networked media. ⟨http://3d-scene.eu/⟩.
[2] C. Aguilera, F. Barrera, F. Lumbreras, A.D. Sappa, R. Toledo, Multispectral image feature points, Sensors 12 (9) (2012) 12661–12672.
[3] E.J. Aguilera-Aguilera, Á.C. Poyato, F.J. Madrid-Cuevas, R.M. Carnicer, The computation of polygonal approximations for 2d contours based on a concavity tree, J. Vis. Commun. Image Represent. 25 (8) (2014) 1905–1917.
[4] C. Akinlar, C. Topal, Edlines: real-time line segment detection by edge drawing (ed), In: Proceedings of the 2011 International Conference on Image Processing, 2011, pp. 2837–2840.
[5] A. Ansar, K. Daniilidis, Linear pose estimation from points or lines, IEEE Trans. Pattern Anal. Mach. Intell. 25 (5) (2003) 578–589.
[6] D.H. Ballard, Generalizing the Hough transform to detect arbitrary shapes, Pattern Recognit. 13 (2) (1987) 111–122.
[7] F. Barrera, F. Lumbreras, A.D. Sappa, Multispectral piecewise planar stereo using Manhattan-world assumption, Pattern Recognit. Lett. 34 (1) (2013) 52–61.
[8] A. Bartoli, P. Sturm, Structure-from-motion using lines: representation, triangulation and bundle adjustment, Comput. Vis. Image Underst. 100 (2005) 416–441.
[9] H. Bay, V. Ferrari, L. Van Gool, Wide-baseline stereo matching with line segments, in: Proceedings of the 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, vol. 1, 2005, pp. 329–336.
[10] J.C. Bazin, Y. Seo, C. Demonceaux, P. Vasseur, K. Ikeuchi, I. Kweon, M. Pollefeys, Globally optimal line clustering and vanishing point estimation in Manhattan world, in: Proceedings of the 2012 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2012, pp. 638–645.
[11] P. Bhowmick, B.B. Bhattacharya, Fast polygonal approximation of digital curves using relaxed straightness properties, IEEE Trans. Pattern Anal. Mach. Intell. 29 (9) (2007) 1590–1602.
[12] M. Brown, Test images for line detection in urban scenes, ⟨http://dx.doi.org/10.15126/surreydata.00807666⟩, 2015.
[13] A.G. Buch, J.B. Jessen, D. Kraft, T.R. Savarimuthu, N. Krüger, Extended 3d line segments from rgb-d data for pose estimation. In: Image Analysis, Lecture Notes in Computer Science, vol. 7944, 2013, pp. 54–65.
[14] J.B. Burns, A.R. Hanson, E.M. Riseman, Extracting straight lines, IEEE Trans. Pattern Anal. Mach. Intell. 8 (4) (1986) 425–455.
[15] J. Canny, A computational approach to edge detection, IEEE Trans. Pattern Anal. Mach. Intell. 8 (6) (1986) 679–698.
[16] D. Ceylan, N.J. Mitra, Y. Zheng, M. Pauly, Coupled structure-from-motion and 3d symmetry detection for urban facades, ACM Trans. Gr. 33 (1) (2014) 2:1–2:15.
[17] C.-H. Chang, Y.-Y. Chuang, A line-structure-preserving approach to image resizing, in: Proceedings of the 2012 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2012, pp. 1075–1082.
[18] T. Chen, Q. Wang, 3d line segment detection for unorganized point clouds from multi-view stereo, in: Proceedings of the 10th Asian Conference on Computer Vision, vol. 2, 2011, pp. 400–411.
[19] W.J. Christmas, J. Kittler, M. Petrou, Structural matching in computer vision using probabilistic relaxation, IEEE Trans. Pattern Anal. Mach. Intell. 17 (8) (1995) 749–764.
[20] P. David, D. DeMenthon, R. Duraiswami, H. Samet, Simultaneous pose and correspondence determination using line features, in: Proceedings of the 2004 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2003, pp. 424–431.
[21] A. Desolneux, L. Moisan, J.-M. Morel, Meaningful alignments, Int. J. Comput. Vis. 40 (1) (2000) 7–23.
[22] D.M. Endres, J.E. Schindelin, A new metric for probability distributions, IEEE Trans. Inf. theory 49 (7) (2003) 1858–1860.
[23] A. Eternadi, Robust segmentation of edge data, in: Proceedings of the 1992 International Conference on Image Processing and Its Applications, 1992, pp. 311–314.
[24] B.J. Frey, D. Dueck, Clustering by passing messages between data points, Science 315 (2007) 972–976.
[25] I. Grosse, P.B. Galván, P. Carpena, R. Román-Roldán, J. Oliver, H.E. Stanley, Analysis of symbolic sequences using the Jensen–Shannon Divergence, Phys. Rev. E 65 (4) (2002).
[26] R. Hartley, A. Zisserman, Multiple View Geometry in Computer Vision, second ed., 2003.
[27] D.C. Hauagge, N. Snavely, Image matching using local symmetry features, in: Proceedings of the 2012 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2012. pp. 206–213.
[28] M. Holtzman-Gazit, L. Zelnik-Manor, I. Yavneh, Salient edges: a multi scale approach, in: European Conference on Computer Vision, Workshop on Vision for Cognitive Tasks, 2010.
[29] M. Hutter, Distribution of mutual information. In: Advances in Neural Information Processing Systems, vol. 14, 2002, pp. 399–406.
[30] J. Illingworth, J. Kittler, A survey of the Hough Transform, Comput. Vis. Gr. Image Process. 44 (1) (1988) 87–116.
[31] L. Itti, C. Koch, A saliency-based search mechanism for overt and covert shifts of visual attention, Vis. Res. 40 (10–12) (2000) 1489–1506.
[32] T. Kadir, M. Brady, Saliency, scale and image description, Int. J. Comput. Vis. 45 (2) (2001) 83–105.
[33] B. Kamgar-Parsi, B. Kamgar-Parsi, Algorithms for matching 3D line sets, IEEE Trans. Pattern Anal. Mach. Intell. 26 (5) (2004) 582–593.
[34] H. Kim, Impart multi-modal dataset. ⟨http://dx.doi.org/10.15126/surreydata.00807707⟩, 2014.
[35] H. Kim, A. Hilton, Influence of colour and feature geometry on multi-modal 3d point clouds data registration, in: Proceedings of the 2014 International Conference on 3D Vision, 2014, pp. 202–209.
[36] M. Klaudiny, M. Tejera, C. Malleson, J.-Y. Guillemaut, A. Hilton, Scene digital cinema datasets, 2014 ⟨http://dx.doi.org/10.15126/surreydata.00807665⟩.
[37] Y. Lin, C. Wang, J. Cheng, B. Chen, F. Jia, Z. Chen, J. Li, Line segment extraction for large scale unorganized point clouds, ISPRS J. Photogramm. Remote Sens. 102 (1) (2015) 172–183.
[38] J.-L. Liu, D.-Z. Feng, Two-dimensional multi-pixel anisotropic Gaussian filter for edge-line segment (els) detection, Image Vis. Comput. 32 (1) (2014) 37–53.
[39] J.G. Lopera, N. Ilhami, P.L. Escamilla, J.M. Aroza, R.R. Roldán, Improved entropic edge-detection, in: International Conference on Image Analysis and Processing, 1999. pp. 180–184.
[40] D.G. Lowe, Distinctive image features from scale-invariant keypoints, Int. J. Comput. Vis. 60 (2) (2004) 91–110.
[41] J. Matas, C. Galambos, J. Kittler, Robust detection of lines using the progressive probabilistic Hough transform, Comput. Vis. Image Underst. 78 (1) (2000) 119–137.
[42] K. Mikolajczyk, T. Tuytelaars, C. Schmid, A. Zisserman, J. Matas, F. Schaffalitzky, T. Kadir, L.V. Gool, A comparison of affine region detectors, Int. J. Comput. Vis. 65 (1–2) (2005) 43–72.
[43] M.T. Parvez, Optimized polygonal approximations through vertex relocations in contour neighborhoods, Image Vis. Comput. 34 (2015) 1–10.
[44] N. Pugeault, F. Wörgötter, N. Krüger, Visual primitives: local, condensed, semantically rich visual descriptors and their applications in robotics, Int. J. Human. Robot. 7 (3) (2010) 379–405.
[45] M.A. Ruzon, C. Tomasi, Edge, junction, and corner detection using color distributions, IEEE Trans. Pattern Anal. Mach. Intell. 23 (11) (2001) 1281–1295.
[46] L. Shao, T. Kadir, M. Brady, Geometric and photometric invariant distinctive regions detection, Inf. Sci. 177 (4) (2007) 1088–1122.
[47] I. Stamos, P.K. Allen, 3-d model construction using range and image data, in: Proceedings of the 2000 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2000, pp. 531–536.
[48] R.G. von Gioi, J. Jakubowicz, J.-M. Morel, G. Randall, On straight line segment detection, J. Math. Imaging Vis. 32 (3) (2008) 313–347.
[49] R.G. von Gioi, J. Jakubowicz, J.-M. Morel, G. Randall, LSD: a fast line segment detector with a false detection control, IEEE Trans. Pattern Anal. Mach. Intell. 32 (4) (2010) 35–55.
[50] F. Wang, B.C. Vemuri, A. Rangarajan, S.J. Eisenschenk, Simultaneous nonrigid registration of multiple point sets and atlas construction, IEEE Trans. Pattern Anal. Mach. Intell. 30 (11) (2008) 2011–2022.
[51] Z. Wang, F. Wu, Z. Hu, Msld: a robust descriptor for line matching, Pattern Recognit. 42 (5) (2009) 941–953.
[52] H. Zeng, X. Deng, Z. Hu, A new normalized method on line-based homography estimation, Pattern Recognit. Lett. 29 (9) (2008) 1236–1244.
[53] L. Zhang, R. Koch, An efficient and robust line segment matching approach based on lbd descriptor and pairwise geometric consistency, J. Vis. Commun. Image Represent. 24 (7) (2013) 794–805.

Mark Brown received a first class honours degree in Mathematics from the University of Bath, UK, in 2012. He is currently a Ph.D. student at the University of Surrey, researching techniques for multi-modal data registration and recognition for digital film production.

David Windridge is a Senior Lecturer in Computer Science at Middlesex University and heads the university's Data Science activities. His research interests centre of the fields of machine learning (particularly kernel methods & classifier-fusion), cognitive systems and computer vision. He has authored more than 80 peer-reviewed publications.

Jean-Yves Guillemaut is a Lecturer in 3D Computer Vision at the University of Surrey. He received the Ph.D. degree in image-based object reconstruction from the University of Surrey in 2005. His research interests include 3D modelling, image/video segmentation and matting, 3D video and animation.